PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-273667
(43)Date of publication of application : 20.10.1995

(51)Int.CI. H03M 7/46
G06F 5/00

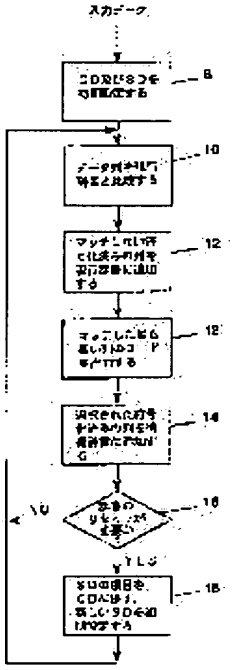
(21)Application number : 07-013347
(22)Date of filing : 31.01.1995

(71)Applicant : HEWLETT PACKARD CO <HP>
(72)Inventor : CLARK II AIRELL R
TOBIN JEFFREY P
SEROUSSI GADIEL

(30)Priority
Priority number : 94 192878 Priority date : 07.02.1994 Priority country : US

(54) DEVICE AND METHOD FOR LEMPEL-ZIV DATA COMPRESSION IMPROVED IN MULTIPLE-DICTIONARY MANAGEMENT IN CONTENT ADDRESSABLE MEMORY

(57)Abstract:
PURPOSE: To facilitate data compression and expansion by using a memory-based dictionary whose lossless type data compressing algorithm has finite size.
CONSTITUTION: For the reduction of loss at the time of data compression due to the resetting of a dictionary, a stand-by dictionary is used to store a subset of encoded data items stored in a current dictionary before. In a 2nd embodiment, data are compressed and expanded according to the address positions of data items included in a dictionary generated in content addressable memory(CAM). In a 3rd embodiment, the minimum memory/high-compressing capacity technique of the stand-by dictionary is combined with a CAM circuit which has a high-speed encoding/decoding function for a single cycle per character. In a 4th embodiment selective overwriting and dictionary switching technique is used and all data items are usable to encode a character string at all times.



LEGAL STATUS

[Date of request for examination] 29.01.2002
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number] 3309028

[Date of registration] 17.05.2002

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平7－273667

(43)公開日　平成７年(1995)10月20日

(51)Int.Cl.*	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｈ０３Ｍ　7/46		8842－5Ｊ		
Ｇ０６Ｆ　5/00		Ｈ		

審査請求　未請求　請求項の数１　ＯＬ　（全 44 頁）

(21)出願番号　特願平7－13347

(22)出願日　平成７年(1995)１月31日

(31)優先権主張番号　１９２８７８

(32)優先日　1994年２月７日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　590000400
ヒューレット・パッカード・カンパニー
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000

(72)発明者　エアレル・アール・クラーク，ザ・セカンド
アメリカ合衆国オレゴン州97330コーヴァリス，ノースイースト・プリマス・サークル・362

(72)発明者　ジェフリー・ピー・トービン
アメリカ合衆国オレゴン州97321アルバニー，サウスイースト・サード・539

(74)代理人　弁理士　古谷　馨　（外２名）

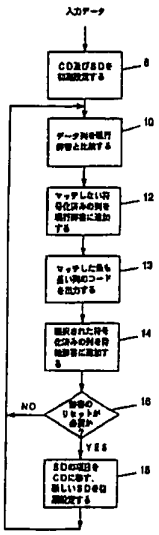
最終頁に続く

(54)【発明の名称】　連想記憶メモリ内の複数辞書管理を改良したＬＥＭＰＥＬ－ＺＩＶデータ圧縮のための装置，及び方法

(57)【要約】

【目的】　ロスレス・タイプのデータ圧縮アルゴリズムが有限サイズを有するメモリ・ベースの辞書を用いて、データ圧縮、及び伸長を容易にすることを目的とする。

【構成】　辞書のリセットによるデータ圧縮の際の損失を低減させるために、待機辞書が、以前、現行辞書に記憶されていた符号化済みデータ項目の部分集合を記憶するために用いられる。本発明の第２の実施態様では、連想記憶メモリ（CAM）内に作成された辞書に含まれるデータ項目のアドレス位置に従って、データが圧縮/伸長される。本発明の第３の実施態様では、待機辞書の最小メモリ/高圧縮能力技法が、文字あたり単一サイクルの高速符号化/復号化機能を有したCAM回路と組み合わせられる。本発明の第４の実施態様では、選択的上書き・辞書切り替え技法が使用され、全データ項目を常に文字列の符号化に使用できる。



【特許請求の範囲】

【請求項１】メモリ・ベースの辞書を使用して、文字列から成るデータを圧縮、及び伸長するための方法において、

複数の記憶位置を含むメモリ装置を提供する方法であって、各記憶位置が文字列に関するコードワードをデータ項目として記憶するための固有のアドレスを有する上記方法と、

メモリ装置の複数の記憶位置内に少なくとも第１、及び第２の辞書を定義する方法と、

文字列に固有に対応するコードワードを各データ項目として記憶する方法と、第１、及び第２の辞書の少なくとも１つに、記憶された各データ項目を割り当てる方法と、

入力データ文字列を表すコードワード値を生成する方法であって、前記コードワードが、入力文字列の一部に対応し、前記辞書の１つに割り当てられ、以前に記憶されたメモリ内のコードワードに関連付けられた前記方法と、

現在辞書の１つに割り当てられているデータ項目の１つを、新しい文字列に関連する新しいコードワードで選択的に上書きし、それによって、データ圧縮、及び伸長処理の間で、常に文字列に関するコードワードを生成するために、第１、及び第２の辞書内のデータ項目全て使用する方法を含むことを特徴とする方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、全般的には、データ圧縮、伸長方法、及びその装置に関し、より詳しくは、辞書を使用して圧縮、及び伸長情報を記憶するロスレス・データ圧縮アルゴリズムの実施に関する。

【０００２】

【従来の技術】圧縮技法の主要なタイプは、バイナリ・シーケンス、又は、他では個々の文字を符号化するために使用されない「コードワード」を使用して複数文字列を符号化する。その文字列は、「アルファベット」、又は単一文字列で構成される。このアルファベットは、コンプレッサが処理する最小の固有情報を表す。従って、８ビットを使用して文字を表すアルゴリズムでは、アルファベットに256個の固有文字を有することになる。圧縮は、所与のファイルのデータ・ストリームにおいて符号化方式で表される複数文字列が出現する程、効果的である。人間の言語間の翻訳のために使用される２か国語辞書から類推して、圧縮前コードと圧縮済みコードの間のマッピングを実施する装置を一般に「辞書」と呼ぶ。

【０００３】一般に、辞書に基づく圧縮技法の有用性は、複数文字列に対応する辞書項目が使用される頻度に依存する。据え付けの辞書が、あるファイル・タイプのために最適化されると、その辞書は他のファイル・タイプには適さないものとなる。例えば、新聞のテキスト・ファイルにあるような多数の文字の組み合わせを含む辞書は、データベース・ファイル、スプレッドシート・ファイル、ビットマップ・グラフィクス・ファイル、コンピュータ援用設計ファイル等を効率的に圧縮することには適さない。

【０００４】所与の入力データが圧縮されている間に、その入力データを圧縮するために使用される辞書が発展する適応圧縮技法が知られている。圧縮前入力データにおける、全ての単一文字を表すコードワードが辞書内に置かれる。そのファイルで、複数文字列に出会うと、辞書に追加項目が加えられる。追加されたその辞書項目は、次にその複数文字列が現れた時に、それを符号化するために使用される。例えば、現時点の入力パターンのマッチングは、現時点で辞書に存在するフレーズに対してしか試みられない。マッチングが失敗するたびに、辞書に新しいフレーズが追加される。その新しいフレーズは、マッチしたフレーズを１つの記号（例えば、マッチングを「妨げる」入力記号）で拡張することによって形成される。圧縮は、辞書が発展する際に、ファイル内で最も頻繁に出現する複数文字列に出会う程、効果的に行われる。

【０００５】伸長の間、辞書は同様の方法で作成される。従って、圧縮済みファイルで文字列のコードワードに出会った時、辞書は、対応する文字列を再作成するために必要な情報を取り込む。広く使用されている、圧縮、及び伸長情報を記憶するための辞書を使用する圧縮アルゴリズムは、それぞれLZ1、及びLZ2と呼ばれる第１、及び第２のLempel-Ziv法である。これらの方法は、Eastman他の米国特許第4,464,650号で開示され、このアルゴリズムに対する様々な改良がWelchの米国特許第4,558,302号、及びMiller他の第4,814,746号で開示されている。これらの参考文献は、更に辞書の使用法を説明している。

【０００６】実際の使用にあたっては、圧縮/伸長に使用できるメモリの量は有限である。従って、辞書内の項目数は有限であり、その項目を符号化するために使用されるコードワードの長さは制限を受ける。通常、コードワードの長さは12ビットから16ビットまで様々である。入力データ・シーケンスが十分長い場合、辞書は最終的に「満杯になる」。この時点では、いくつかの動作が選択可能である。例えば、辞書をその状態で凍結させ、入力シーケンスの残りの部分に対して使用することができる。第２のアプローチでは、辞書がリセットされ、新しい辞書が最初から作成される。第３のアプローチでは、圧縮率が低下するまで、しばらくの間辞書を凍結させ、その後リセットする。

【０００７】第１の代替案は、基本圧縮アルゴリズムの学習能力が失われる欠点がある。入力データの統計が変化した場合、その辞書はもはやこれらの変化に追随できず、圧縮率の急激な低下が発生する。

【０００８】辞書をリセットする方法は、アルゴリズムの学習能力を維持するが、空の辞書に切り替えた時に、一時的に圧縮率が低下する(例えば、以前に異領されたそのソースの知識が全て失われる)。例えば、リセット時には、全ての辞書項目が無差別に使用不能となる。従って、最近得られた、以降のデータ圧縮で使用されるであろう辞書項目は、以降、圧縮、及び伸長処理の助けとなる確率が低い、より古いデータ項目と共に失われる。辞書のリセット時に全てのデータ項目が失われるので、圧縮率は一時的に低下する可能性がある。従って、圧縮効率は最適圧縮効率より低い。

【０００９】辞書ベースのデータ圧縮効率を向上させるための１つの方法は、Bunton、及びBorriello等「PRACTICAL DICTIONARY MANAGEMENT FOR HARDWARE DATA COMPRESSION」(Communications of the ACM, 1992年１月、第35巻、第１号)で論じられている。全辞書のリセットは、一度に１つの辞書項目を置換することによって回避できる。LRUコードが選択され、次の入力文字列で上書きされる。Bunton他によるこの方法は、圧縮率を改良させたが、LRU状況を識別するために各辞書項目ごとに多数の追加ビットを必要とする欠点がある。各辞書項目ごとの追加ビットによって、ハードウェアのコストが大幅に増加する。

【００１０】必要とされる辞書のリセット回数を減らす１つの方法は、辞書のメモリ・サイズを大きくすることである。しかし、メモリ・サイズを大きくすると、コストが増加し、辞書のデータ項目を探索するために必要な時間を増大させる可能性がある。更に、現在のLRU記録方法は、大きなメモリ・サイズにおいては実際的ではなくなっている。

【００１１】圧縮/伸長性能に関する別のボトルネックは、以前に出現した文字列を辞書で探索するのに必要な時間の量である。従来、ハッシング・アルゴリズムを使用して、以前に記憶された辞書項目が探索されて、新しい文字列が使用できるメモリ位置が見付けられた。典型的な構成は、Welchの米国特許第4,558,302号(LZW)で開示されたように、各辞書項目ごとに２つないし４つの記憶位置を有するRAMメモリを使用する。

【００１２】ハッシング・アルゴリズムは、ある簡単なデータ・ワード内容の数学関数に基づくRAM空間のアドレスに、固有の各辞書項目をマップする。そのようなアルゴリズムは、ワード全体、又はワード内のフィールドを使用して、マッピング・アドレスを計算するので、メモリ内の同じ位置に複数のデータ・ワードがマップされ、ハッシング衝突が発生する可能性がある。この場合、そのデータ用に代替位置を見付けなければならない。RAMの記憶場所が満杯になると、必然的に、第２の辞書項目は、以前に使用された位置にハッシュされる。この状況は圧縮が継続可能なうちに、解決されなければならない。ハッシング回路、及び特にハッシング衝突によって、圧縮/伸長システムの論理機構にかなりの複雑さが加わり、システムのスループットが低減する。

【００１３】通常、圧縮されるデータに基づく辞書は、全てのデータ項目のうちのわずかな部分集合である。従って、ハッシング衝突を低減する１つの方法は、辞書記憶位置の数を増やすことである。しかし、このアプローチは、システムの複雑さ、及びコストを増大させ、メモリと圧縮/伸長を制御する論理機構との統合を妨げる。更に、より大きなメモリでは、文字列が以前にメモリ内にロードされたかどうかを判定するために必要な探索時間が増加する恐れがある。

【００１４】データ圧縮/伸長に関する別のボトルネックは、データ文字列を符号化、及び復号化するために必要な時間の量、及び回路の複雑さである。例えば、データ圧縮の間、文字列が、以前にメモリ内に記憶されたどのデータ・フレーズともマッチしないことが分かった後で、占有されていないデータ・メモリ位置に記憶されなければならない。その記憶された文字列、及び以前に辞書のデータ項目にマッチした文字列のサブフレーズを固有に識別するコードワードが生成されなければならない。次に、そのコードワードは、以降のデータ圧縮動作時に追加文字と組み合わせが可能となるように記憶されなければならない。

【００１５】データ伸長の間、圧縮済みデータのコードワードは、Hewlett-Packard Journal (1989年６月、27ないし31ページ)に記載されているように、圧縮前データ文字、及び追加コードワード、例えば圧縮前データの列の残りとのリンクを表すことができる。ここで記載されたHP-DC技法は、コードワードを順次に符号化し、圧縮済みコードによって決定される辞書のアドレス位置に、次のバイト(K)と連結されたコードワード(OMEGA)を記憶する。従って、実際の伸長済みデータの列が生成される前に、辞書を数回読む必要がある。圧縮、及び伸長処理が反復的なものなので、辞書のアクセスに使用されるクロック・サイクル以外のあらゆる追加クロック・サイクルによって全体の圧縮、及び伸長時間が大幅に増大する。しかし、現在の符号化、復号化、及び辞書探索方法では、各入力文字を圧縮、又は伸長するのに２回以上のクロック・サイクルが必要である。更に、これらの符号化、及び復号化アルゴリズムには、複雑な圧縮、及び伸長ハードウェアが必要となる。

【００１６】従って、辞書ベースのデータ圧縮システムの性能を改良し、辞書ベースのデータ圧縮/伸長システムにおけるデータの符号化、及び復号化を改良する必要がある。

【００１７】

【発明が解決しようとする課題】従って、本発明の課題は、辞書ベースの圧縮システムにおいて、辞書がリセットされた時に被るデータ圧縮の損失を最小限に抑えることである。

【００１８】本発明の第２の課題は、統計的特性が変化する入力データ・シーケンスに関するデータ圧縮システムの適応特性を向上させることである。

【００１９】本発明の別の課題は、辞書ベースのデータ圧縮/伸長システムにおいて、文字列を符号化/復号化するために必要な時間の量を低減させることである。

【００２０】本発明の別の課題は、最小量のメモリを有する、辞書ベースのデータ圧縮/伸長システムにおいて、データ圧縮能力を最大にすることである。

【００２１】本発明の別の課題は、辞書ベースのデータ圧縮/伸長システムを選択的に更新するために必要とされるハードウェア、及び時間の量を最小限に抑えることである。

【００２２】

【課題を解決するための手段】本発明の１つの実施態様は、現行辞書、及び待機辞書を同時に作成するデータ圧縮/伸長システムである。現行辞書は、標準データ圧縮エンジンの辞書と同じ目的を果たす。待機辞書は、現行辞書のフレーズの部分集合を含むように、現行辞書と並行的に作成される。このサブセットは、ソース・データ内で出現するパターンを最もよく特徴付けるように選択される。現行辞書が満杯になると、待機辞書と置換され、新しい待機辞書の作成が継続され、かつ圧縮に使用される新しい現行辞書として、「最初から」作成される。従って、コンプレッサは決して空の辞書に切り替えることなく、限られた辞書のメモリ・サイズを有することによって発生するデータ圧縮率の低下が抑えられる。

【００２３】この現行辞書は、新しいデータ項目を追加するのに十分な空の空間を有した状態で始まり、それによって、ソース・データへの適応を継続することが可能となる。この特徴は、様々な統計を有するソース・データを圧縮するうえで極めて重要なものである。より少ない数のデータ項目を有する待機辞書に切り替わることによって、幾つかの情報が失われるが、辞書を最大効率になるよう再作成する時間は、辞書を完全にリセットするよりずっと短い。従って、データ圧縮率に対する悪影響を小さくして、より小さい辞書メモリを使用することができる。

【００２４】待機辞書に格納される現行辞書の部分集合を選択する基準は、特定のアプリケーションに応じて変更することができる。例えば、符号化済みデータ列は、現行辞書のデータ項目に少なくとも１回マッチした場合、待機辞書にコピーされる。又、待機辞書の項目は、列の長さ、最近のデータ項目のマッチ、又は所与のアプリケーションで圧縮を最大にする項目を識別する任意の基準に応じて選択することができる。

【００２５】更に、現行辞書から待機辞書に切り替える(リセットする)ための基準は、データ、又はアプリケーションのタイプに応じて変更することができる。例えば、現行辞書は、有効なデータ項目で満杯になった時にリセットすることができる。代替案として、Kato他の米国特許第4,847,619号に記載されたように、現行辞書を、圧縮に使用して、所定の性能閾値より低くなった時にリセットすることができる。

【００２６】主としてデータ特性に変化のない状況での待機辞書の第２の応用例では、コンプレッサはデータに対して２パスを行う。第１パスで、コンプレッサはそのデータの大きなサンプルを走査する。そのサンプルは、現行辞書を何度も満杯にさせるのに十分な大きさであり、それによって、待機辞書はサンプルの大きさに対応する回数分だけ現行辞書と置換される。数回繰り返して、アルゴリズムによって、データ・サンプルに対して強力にカスタマイズされた辞書が作成されるまで、辞書が切り替わるたびに現行辞書が「洗練」される。次に、カスタマイズされた辞書は、第２パスの間に、入力データを圧縮するために圧縮エンジンによって使用される、単一参照辞書としてセットされる。それによって、カスタマイズされた辞書は、同じデータに対する単一の動的辞書よりもかなり高い性能を示す。

【００２７】本発明の第２の態様は、圧縮/伸長システムの辞書に記憶されたデータ項目のアドレス値を使用して、符号化回路と復号化回路を簡略化する、辞書ベースの圧縮/伸長システムのアーキテクチャ、及び方法である。このシステムは、ローカル・フィードバック回路を含む追加論理回路を有した連想記憶メモリ(CAM)を使用して、メモリ・アクセスを高速化し、外部圧縮/伸長の論理機構を簡略化する特殊機能を提供することが好ましい。このメモリ構造は、ハッシングやハッシング衝突の可能性がなく、１クロック・サイクル当たりに１文字という一様の速度で、ロスレスでデータ圧縮、又は伸長できる固有の特徴を有する。

【００２８】特に、このシステムは、メモリ内に含まれるデータ項目のアドレス位置に応じて文字列を符号化するアソシエイティブ・メモリを備えることが好ましい。以前に入力データ・ストリーム内で出現しなかった入力文字列の組み合わせが、辞書内の新しいデータ項目として記憶される。そのCAMは、それぞれが固有の文字列データ項目を記憶する、「ワード」として組織化される。メモリは、以前辞書に記憶されていた全てのワードに対して、「ワード」内から選択されたビットを有する入力文字列で、アソシエイティブ並行探索を実施する。マッチした場合、そのデータ項目に関連するマッチ・ラインが活動化される。次に、全てのマッチ・ラインが、その文字列を表す単一のコードワードに符号化される。次に、そのコードワードは、次の入力文字と組み合わせられ、再度、以前メモリに記憶されたデータ項目と比較される。従って、メモリ内の文字列のアドレス位置に応じて前記文字列にコードワードが割り当てられる。探索が失敗すると、最後にマッチした文字列を表すコードワード(OMEGA)(前記文字列のアドレス)が出力され、マッチを失敗

させた文字(K)で始まる新しい文字列で別の探索が開始される。圧縮済みデータ文字(コードワード)は、辞書内のデータ項目へのポインタである。従って、文字列は、圧縮済みデータ文字を伸長用の辞書へのアドレスとして使用することによって復号化される。例えば、最初に外部圧縮済み文字が辞書へのアドレスとして使用される。次に、復号化済みアドレス位置におけるデータ項目が読み取られる。メモリから出力されたデータ項目が、これ以上の伸長を必要としない(例えば、メモリ出力が、リンク・リストの「ルート」である)場合、そのデータ項目は出力される。データ項目が他のコードワード(例えば、更に符号化された別の辞書のアドレス位置へのリンク)を含む場合、そのアドレスにおける文字が出力され、そのアドレスにおけるコードワードが次の辞書アドレスとしてメモリにフィードバックされる。

【0029】内部アドレス・ジェネレータは、圧縮、及び伸長の両方に使用され、メモリのリセットと同時にリセットされる。メモリへのあらゆる書き込み(明示的な書き込み、又は失敗したマッチング)の結果、アドレスは次のアドレスにインクリメントされる。インクリメントは順次である必要はないが、コンプレッサ、及びデコンプレッサ・アドレス・ジェネレータが両方同じ状態に初期設定され、同じようにインクリメントした結果、圧縮用の辞書と伸長用の辞書が両方同じものとなるかぎり、例えば擬似ランダムであってよい。この論理機構によって、外部制御論理機構でアドレスを生成/記憶する必要がなくなり、圧縮、及び伸長の性能が向上する可能性がある(例えば、クロック・サイクルが減少し、データ圧縮が高速になる)。

【0030】データ圧縮に必要な時間を更に減らすために、特殊更新回路によってメモリ探索、及びデータ書き込みを同一クロック・サイクル中に実行することができる。文字列がメモリ内のデータ項目と比較される時、失敗した探索は、新しいデータ項目として記憶されるべき文字列を必要とする。次の利用可能なアドレス位置は既に、アドレス・ジェネレータによって分かっており、文字列は既にメモリ・データの入力時点で存在している。従って、探索の間、1つもマッチしない場合、制御論理機構を使用して文字列を自動的にメモリに書き込むことができる。従って、メモリは、メモリ探索のクロック・サイクル中に自動的に更新される。探索動作時にマッチがあった場合、この更新回路は文字列が有効なデータ項目としてメモリにロードされるの防ぐ。

【0031】それによって、上記で要約したシステム、及び方法は、データを高速に圧縮、及び伸長するための簡単で、廉価で、かつ多機能なシステムを提供する。このシステム、及び方法は、汎用コンピュータ上のソフトウェアで実施することも、あるいはカスタム、又はセミカスタム集積回路を使用するハードウェアで実施することも可能である。このシステム、及び方法を使用して、リンク・リストのデータ構造の記憶/検索を実施することができる。更に、このシステム、及び方法は、様々な適応辞書ベースのエンコーダに容易に適応させることができる。

【0032】本発明の第3の態様は、待機辞書の最小メモリ/高圧縮能力技法を文字あたり単一サイクルの高速符号化/復号化機能を有したCAM回路と組み合わせる。この回路は、CAM回路の記憶位置内で複数の辞書を使用する。CAM回路は、圧縮済み文字列、及び圧縮前文字列を受け取り、それらを辞書の1つにデータ項目として記憶する。次に、文字列にマッチする辞書のデータ項目のアドレスに応じて、各データ文字列を表すコードワードが生成される。

【0033】複数の辞書をサポートするために、CAM内の各メモリ位置は、状況フィールドとデータ・フィールドを含む。データ・フィールドは、データ項目を記憶し、状況フィールドは、どの辞書がそのデータ項目に割り当てられているかを示している。探索動作の間に、この回路は状況フィールドとデータ・フィールドの両方のあるビットをマスクすることができる。このことによって、このシステムは、どの辞書がデータ項目に割り当てられているかを判定し、あるメモリ位置が現在、辞書に割り当てられていないかどうかを判定できる。

【0034】各データ項目ごとの辞書の割り当ては、圧縮/伸長回路の状態を変更することによって容易に切り替えられる。回路の状態を変更することによって、少なくとも1つの辞書がリセットされる。このことによって、以前その辞書に割り当てられていた記憶位置は、今度は、もはやどの辞書にも割り当てられていないフリー記憶位置を構成することができる。このようなフリー記憶位置は今度は、新しい文字列を記憶するために使用することができる。圧縮率、及び異なるデータ・タイプへのシステムの適応性を最大にするために、異なる事象をきっかけとして状態の変更を行うことができる。例えば、圧縮/伸長回路は、辞書の1つが満杯になった時に自動的に状態を変更することも、圧縮率が所定の性能レベルより低くなった時に状態を変更することもできる。

【0035】圧縮/伸長システムの圧縮率を更に増加させるには、第2のLempel-Ziv圧縮/伸長システム(LZSD2)を使用して、個別のデータ項目を新しい文字列と選択的に置換する。LZSD2優先順位システムは、文字列マッチングのために全ての辞書の項目を常に使用できるが、それでも上述の待機辞書方式を使用している。従って、辞書のサイズにかかわらず、次の上書き位置を識別するには、2ビットだけが必要とされる。この場合、各データ項目が辞書のリセット後も辞書に割り当てられたままなので、辞書は、データ圧縮率に影響を与えずに更新されうる。データ圧縮率に悪影響を与えずに、上述したのと同じ圧縮/伸長ハードウェアを用いて実施することも可能である。

【0036】単一クロック・サイクル探索機能を提供するために、圧縮/伸長回路は、現行辞書と並行的に待機辞書を作成し、複数の辞書を同時に探索する。

【0037】本発明の前記、及び他の目的、特徴、及び利点は、図面に関連して進める好適実施例に関する以下の詳細な説明から容易に明らかになろう。

【0038】

【実施例】以下の説明では、第1節、及び第2節はそれぞれ、本発明の待機辞書、及び連想記憶メモリの態様を説明する。第3節は、本発明の最初の2つの態様を組み合わせた実施態様を説明する。第4節は、第3節で説明したシステムを使用する、代替動作方法を説明する。

【0039】1. 待機辞書を使用するデータ圧縮/伸長システム

図1は現行辞書と待機辞書を用いたデータ圧縮/伸長システムのデータ・フロー図である。図1に示した方法は現行辞書(CD)、及び待機辞書(SD)の両方を初期設定するブロック8から始まる。例えば、圧縮前入力データ内の全ての単一文字を表すコードワードが辞書内に置かれる。又、最初の辞書は空であってもよい。データ・シーケンスからの文字列の符号化は、任意の所望の符号化技法を使用して実施される。

【0040】ブロック10で、入力データが、以前に符号化された現行辞書のデータ項目と比較され、その文字列と、辞書のデータ項目のどれかがマッチするかどうか判定される。ブロック12で、マッチしない文字列が、新しい符号化済みデータ項目として現行辞書に記憶される。もはやマッチングが拡張できなくなると、ブロック13で、マッチした最も長い文字列のコードが出力される。

【0041】ブロック14で、現行辞書(CD)の以前に符号化されたデータ項目の部分集合が待機辞書(SD)に記憶される。ブロック14における上述の部分集合選択処理は、特定の入力データに関して、待機辞書内の所与の数のデータ項目を用いて最大の圧縮率をもたらすように修正することができる。例えば、待機辞書のデータ項目は、入力文字列が辞書内のデータ項目にマッチする回数に基づいて選択されうる。又、待機辞書の部分集合は、符号化済みの文字列によって表現される入力文字の数に応じて選択されうる。一般に、この段階で、好適ないかなる技法も適用することができる。

【0042】決定ブロック16で、辞書のリセットが必要かどうかが判定される。例えば、現行辞書において符号化済み文字列の項目数が所定数に達した時、あるいは圧縮率が所与の性能閾値より低くなった時、リセットが必要とされる。現行辞書をリセットする必要がない場合、圧縮エンジンは新しい文字列を読み取り、処理はブロック10に戻る。現行辞書がリセットされた場合、ブロック18は、現行辞書を待機辞書内の項目と置換し、新しい待機辞書を初期設定し、新しい文字列を読み取り、次いでブロック10に戻る。

【0043】図1による辞書ベースの圧縮/伸長方法を使用して、データを圧縮するために使用される静的にカスタマイズされた現行辞書を生成することができる。例えば、入力データ・シーケンスのデータ・サンプルが選択される。次に、現行辞書を繰り返し待機辞書と置換することによって、現行辞書がカスタマイズされる。カスタマイズされた現行辞書は次に、読み取り専用機能にロックされ、データ・シーケンスの圧縮、又は伸長のために圧縮エンジンによって排他的に使用される。

【0044】図2は、現行辞書、及び待機辞書を使用するデータ圧縮アルゴリズムの一例を示す詳細データ・フロー図である。図2は、入力データ列が、現行辞書内の項目とマッチした時に、待機辞書にコピーされる方法を示す。この手順では、データ列は入力において少なくとも2回は確実に「見られている」。現行辞書が満杯になると(例えば、有効なデータ項目が所定数に達した時)、現行辞書と待機辞書が切り替えられる。上述のように、代替辞書切り替え基準、及び待機辞書のデータ項目選択基準は、特定のアプリケーション要件に応じて容易に与えられる。

【0045】ブロック20で、入力データ列が現行辞書のデータ項目と比較される。決定ブロック22は、入力データと現行辞書内の項目の間にマッチがない場合、ブロック23、及び24に分岐する。次にブロック23で、マッチした最も長いデータ列が符号化され、出力される。

【0046】決定ブロック24で、現行辞書が満杯かどうかが判定される。現行辞書が満杯でない場合、ブロック28で、データ列がデータ項目として現行辞書に記憶される。現行辞書が満杯である場合、ブロック26で、現行辞書が待機辞書と切り替えられる。以降、データ列は新しい現行辞書に記憶される。この場合、現行辞書はデータ項目(例えば、待機辞書のデータ項目)の、より小さな部分集合と置換されているので、新しいデータ列を記憶するために使用可能な空間がある。ブロック30で、現行辞書のアドレス・カウンタがインクリメントされる。ブロック34で、入力データから新しい入力文字が読み取られ、次にブロック20の比較処理が繰り返される。

【0047】決定ブロック22で、入力データと現行辞書内の項目の間にマッチがあると判定されると、決定ブロック36で、そのデータ列が以前に待機辞書に記憶されたかどうかを判定するための検査が行われる。そのデータ列が以前に待機辞書にコピーされていなかった場合、現行辞書内の状況フィールドにフラグがセットされる。

又、ブロック36を除く任意の箇所で、このフラグをセットすることができる。このフラグは、そのデータ列にマッチした現行辞書のデータ項目に関連付けられる。このフラグは、このデータ項目が以前に待機辞書にコピーされたことを圧縮エンジンに示す。このことは、同じデータ項目が待機辞書に複数回コピーされるのを防ぐ。ブロック40でデータ列が待機辞書に書き込まれ、ブロック42

で待機辞書のアドレス・カウンタがインクリメントされる。そのデータ列が現行辞書内のデータ項目にマッチしたので、ブロック44で現在のデータ列に次の入力文字が追加され、ブロック20に戻る。決定ブロック36でデータ列が以前に待機辞書に記憶されていたことが示された場合、この処理は直接、ブロック44に進み、上述のように継続される。

【００４８】図３は、現在の好適実施例である、データ・コンプレッサ/デコンプレッサ集積回路(IC)50における本発明の実施例を示している。データ・コンプレッサ/デコンプレッサ(DCD)IC50は、データ・コンプレッサ・インタフェイス回路54と組み合わされたデータ圧縮/伸長エンジン52を含む。DCD IC50は、ランダム・アクセス・メモリ(RAM)88内に構成された辞書１(D1)、及びRAM90内に構成された辞書２(D2)と組み合わせて使用される。本明細書に示した回路は好適には単一のICで、又は別々のIC50,88,及び90として実施される。D1、及びD2はRAMとして示されているが、好適には連想記憶メモリ、又は任意の代替メモリ構造で実施することができる。このRAMは従来型のものである。D1、及びD2内の各RAM辞書メモリ位置はそれぞれデータ項目フィールド(data_entry)94、及び98と、待機状況フィールド(stdby_stat)92、及び96を含む。

【００４９】データ項目フィールドは、入力データ・シーケンスで現れる固有のデータ列を記憶する。待機状況フィールドは、現行辞書内のデータ項目が以前に待機辞書に記憶されていたかどうかを示す待機辞書状況フラグを含む。待機状況フィールドは好適には、有効なデータ項目を識別するためのデータ有効(data_valid)フィールドを含むことができる。データ圧縮システムにおいて複数ビットのデータ有効フィールドを使用することは、1991年９月25日に出願され、本出願人に譲渡された米国特許出願第07/766,475号「DICTIONARY RESET PERFORMANCE ENHANCEMENT FOR DATA COMPRESSION APPLICATIONS」に記述されており、ここで参照することにより、本明細書に組み込まれている。

【００５０】データ圧縮エンジン52は、LZ2、又はLZ1圧縮アルゴリズムを実施するように設計されていることが好ましいが、任意の適当な辞書ベースの圧縮技法を実施するように設計されることも可能である。又、必要であれば、圧縮エンジンは米国特許第4,847,619号で開示されたような、辞書のリセットを制御するための自動手段を組み込むことも、又はその手段と共に使用することもできる。その他の点については従来通りなので、データ圧縮エンジンの特定のアルゴリズム、及びアーキテクチャをより詳細に説明する必要はない。

【００５１】データ・コンプレッサ・インタフェイス回路54は、２つの主要分岐回路を備えている。切り替え制御装置分岐回路76は、データ圧縮エンジン52、又はデータ・コンプレッサ/デコンプレッサICに関連するその他の回路から出力される辞書リセット要求信号72、及びデータ列マッチ信号70を監視する。分岐回路76は、どちらのRAMが現行辞書として動作し、どちらのRAMが待機辞書として動作するかを制御する。この分岐回路は、現行辞書から待機状況フィールドを読み取り、現在の符号化済みデータ項目が以前に待機辞書にコピーされていたかどうかを判定する。

【００５２】アドレス・ジェネレータ回路68は、待機モードで動作する辞書に関するデータ項目フィールドのバイナリ値を順番に並べる。この分岐回路の典型的な実施態様はバイナリ・カウンタであるが、他の形態のシーケンサも容易に使用することができる。この分岐回路には、マルチプレクサ56、58、及びトランシーバ84、86が関連付けられている。マルチプレクサ56は、データ圧縮エンジン52、及び切り替え制御装置回路76のそれぞれからの読み取り/書き込み信号60、及び62の一方を選択する。マルチプレクサ58は、データ圧縮エンジン52、及びアドレス・ジェネレータ68のそれぞれからのアドレス信号64、66の一方を選択する。トランシーバ86は、データ圧縮エンジン52に接続されたデータ・バス87に接続して、データ項目フィールド94、98の一方を選択するバス制御装置として動作する。トランシーバ84は、切り替え制御装置76に接続して、待機状況フィールド92、96の一方を選択する。マルチプレクサ、及びトランシーバは制御信号78によって制御され、アドレス・ジェネレータ68は制御信号82によって制御される。両方の制御信号は切り替え制御装置回路76からのものである。

【００５３】DCD回路50は、通常の圧縮/伸長動作の間に、辞書の１つ(現行辞書)とデータ圧縮エンジン52の間の従来のデータ転送を可能にする。このシステムにおいて、待機辞書は、本発明に従うデータ項目の部分集合を作成するために、データ圧縮エンジン52からデータを、又は現行辞書から直接データを受け取ることもできる。

【００５４】回路50の動作時に、切り替え制御装置76は、現行辞書としてD1、D2の一方、例えばD1を選択する。次に、圧縮エンジンは、データ圧縮の実行、符号化済みデータの読み取り、及びD1のデータ項目フィールド94への符号化済みデータの書き込みを開始する。符号化済みデータ列が待機辞書(D2)への書き込みの候補であると圧縮アルゴリズムが判定した時、例えば、符号化済みデータ列が現行辞書(D1)内のデータ項目とマッチした時、マッチ信号70が活動化される。それによって、切り替え制御装置76は、現行辞書からの待機状況フィールド92を検査して、そのデータ項目が以前に待機辞書にコピーされていたかどうかを判定する。そうでない場合、データ列は、アドレス・ジェネレータ68によって提示される位置における、D2のデータ項目フィールド98内に書き込まれる。更に、現行辞書内の待機状況フィールド92は、同じ符号化済みデータ列が待機辞書に２回コピーされるのを防ぐように、切り替え制御装置76によって「セット」される。

【００５５】データ圧縮エンジン52がリセット信号72を活動化すると、切り替え制御装置76は制御信号78の値を変更する。この新しい制御信号は、D2が次に現行辞書として動作し、D1が次に待機辞書として動作するように、マルチプレクサ、及びトランシーバに関する接続を変更する。次に、D2にロードされているデータ項目の部分集合が、圧縮エンジン52に対するデータ項目の初期セットとして使用される。従って、図３に示したデータ圧縮エンジンがリセットされると、新しい辞書のデータ項目フィールドは、以前に符号化された入力データの高圧縮率の部分集合を含む。

【００５６】切り替え制御装置76は、マッチ信号70、及びリセット信号72の特定の組み合わせを活動化することにより、データ圧縮エンジンによって遮断されうる。これによって、データ圧縮エンジンは単一の辞書から符号化済みデータを排他的に読み取り、単一の辞書に符号化済みデータを排他的に書き込むことができる。これは、上述のカスタマイズされたデータ辞書の動作と、単一辞書技法との互換性のために使用される。

【００５７】上述の方法は、有効であることが分かっている。例えば、標準UNIXの「compress」コマンド(従来型のLZ2実施態様)を使用して、ユーザ操作マニュアルを含む550個のファイルが圧縮されている。次に、上述の現行/待機辞書方式を使用して、カスタマイズされた辞書を作成し、次に、そのファイルがデータ・サンプルから作成したカスタマイズされた辞書を使用して圧縮されている。その結果を以下に要約する。

最初のファイル・サイズ	: 6,602,300バイト
Unix圧縮	: 2,781,686バイト
カスタマイズされた辞書による圧縮	: 2,025,742バイト
圧縮向上比	: 37％

従って、カスタマイズされた辞書を用いる圧縮方法は、従来型の圧縮方法より実質的な圧縮率が改良されている。

【００５８】本発明のこの態様は、その基本的原理から逸脱することなく構成、及び細部に関して修正することができる。例えば、項目が待機辞書にあるかどうかを示すフィールドを有することによって、現行辞書と待機辞書の両方を同一RAM上で実施することができる。リセット時には、待機辞書でない辞書の項目全てがクリアされる。アドレス・ジェネレータ回路は、リセット後に項目が連続位置に位置しなくなるので、より複雑になる。このアプローチは、後述の連想記憶メモリ(CAM)の実施態様に特に適している。

【００５９】II. ロスレスのデータ圧縮/伸長辞書記憶用メモリ回路

図４は、本発明の第２の態様によるCAM圧縮/伸長システム用の回路136の全体構成を示すブロック図である。回路136は、データ圧縮/伸長(CD)エンジン142、圧縮前データ・インタフェイス138、圧縮済みデータ・インタフェイス148、及びプロセッサ・インタフェイス152を含む。CDエンジン142は、列テーブル・メモリ144、及び制御論理機構146を備えている。圧縮前データ・インタフェイス138は、データ・バス154を介して圧縮前データ(RAWDATA)を転送し、圧縮済みデータ・インタフェイス148は、データ・バス158を介して圧縮済みデータ(COMPDATA)を転送する。インタフェイス138、及び148に関する外部制御信号はそれぞれ、制御バス156、及び160を介して受け取られる。各インタフェイスは、先入れ先出しデータ・バッファ(FIFO)140,150と、更に従来型インタフェイス回路(図示せず)を含む。

【００６０】回路136は、圧縮、又は伸長モード/状態のどちらかで使用することができ、双方向通信モードの間で切り替えることができる。又、回路136は、図５のブロック182,184,192,及び194を置き換えて、RAMを用いた簡単な専用伸長回路を含む専用コンプレッサとして使用することができる。以下の説明は、圧縮と伸長の両方に回路136を使用すると仮定している。

【００６１】圧縮モードでは、圧縮前データ・インタフェイス138が、データ・バス154から圧縮前データ文字を受け取り、データ・バッファ140を介してそれを圧縮/伸長エンジン142に供給する。CDエンジン142内の列テーブル・メモリ144、及び制御論理機構146は、データ・バッファ150を介してデータ・バス158上に出力されるコードワードにその文字を圧縮する。伸長モードでは、圧縮済みデータ・インタフェイス148が、データ・バス158から圧縮済みデータ・コードワードを受け取り、データ・バッファ150を介してCDエンジン142にそれを提供する。列テーブル144は制御論理機構146と協働して、そのデータ・コードワードを文字列に伸長し、データ・バッファ140を介してデータ・バス154上にその結果を出力する。マイクロプロセッサ(図示せず)は、データ・フローの方向と圧縮/伸長モードをセットするためのレジスタを制御し、かつプロセッサ・インタフェイス152を介してその他の様々な機能を制御する。

【００６２】図５は、図４の列テーブル・メモリ144、及び制御論理機構146の詳細ブロック図である。列テーブル・メモリは、圧縮/伸長処理時間を減らす追加の内部論理機構を備えた連想記憶RAM(CAM)188形式のアソシエイティブ・アレイから成る。CAM188は、それぞれが別々の文字列項目を記憶する、「ワード」によって組織化される(例えば3832x20ビット)。データは、データ・バス190(DATA_IN)上でメモリ188に書き込まれる。データ・バス190は、バス180上で入力された外部文字列(K)を受け取り、マルチプレクサ192(MUX3)を介してデータ・バス202上で符号化済み文字列(OMEGA)を受け取る。バス180上の外部文字は、RAWDATAバス154(図４)上の圧縮前データ・ストリームから得られ、そのコードワードはエンコーダ194の出力から得られる。データ入力選択論理回路182はマ

ルチプレクサ192を介して、バス180上の外部文字列から得られたDATA_INのビット、及びバス202上のコードワードから得られたDATA_INのビットを制御する。このデータ入力選択論理回路182は、共に制御論理機構146(図4)からの探索信号入力178、及び読み取り/書き込み信号入力164と、エンコーダ194からのマッチ信号入力168によって駆動される。

【0063】メモリ188は、マッチ・ライン206(例えば、264ないし4095)を介してマッチ信号の集合を提供する。メモリ188内の各ワードごとに1つのマッチ信号が関連付けられている。バス190上の文字列がメモリ188内のデータ項目の1つにマッチした時、そのメモリ位置に関連するマッチ信号が活動化される。エンコーダ194は、メモリ188からの全てのマッチ信号を符号化し、順に、バス202上で提供されるコードワードを生成する。それによって、コードワードは、メモリ188内のマッチしたデータ項目のアドレス位置に等しくなる。エンコーダ194は、メモリ188内のデータ項目のどれかがデータ・バス190上の文字列にマッチした時に活動化されるマッチ信号168も生成する。

【0064】アドレス・デコーダ184は、外部アドレス・バス177を介して外部圧縮済み文字、データ・バス186上でメモリ188から出力された内部文字列、又はアドレス・ジェネレータ170からの内部アドレスのいずれかを選択的に受け取り、ワード選択ライン204(例えば、264ないし4095)を介してアソシエイティブ・アレイ188にアクセスする。バス177上の外部圧縮済み文字は、COMPDATAバス158(図4)上の圧縮済みデータ・ストリームから得られる。内部アドレス・ジェネレータ170は、エンコーダ194からのマッチ信号168、探索信号178、読み取り/書き込み信号164、及びリセット信号162によって制御される。読み取り/書き込み、及びリセット信号は、制御論理機構146(図4)から得られる。このアドレス・ジェネレータは、初期設定時に(例えば264に)リセットされ、続いて辞書が作成される時にインクリメントされるカウンタを含む。

【0065】アドレス・デコーダ184に供給されるアドレスのソースは、それぞれマルチプレクサ176、174を介して、読み取り選択論理機構172、及び読み取り/書き込み信号164によって制御される。読み取り選択論理機構172は、リセット信号162、及びメモリ188から出力されるデータ項目186の圧縮状況によって制御される。このデータ項目の圧縮状況は、データ項目文字の値によって判定されうる。例えば、256より大きな値は符号化済み文字列に割り振られることができ、256より小さい値は単一データ文字を構成することができる。マルチプレクサ176(MUX1)は、バス177、又はバス186から入力を選択し、マルチプレクサ174(MUX2)はマルチプレクサ176の出力とアドレス・ジェネレータ170の出力から一方を選択する。デコーダ184は、データ探索、及びメモリ更新を同じメモリ・アクセス・サイクルで実行できる、以下で説明する自動更新機能を含む。

【0066】図6は、図5のアドレス・デコーダ184の自動更新機能の好適実施例の論理図である。マルチプレクサ174(MUX2)からアドレス・デコーダ184に入力された各アドレス(ADDRN[19:0])は、2つのANDゲートに送られる。ANDゲート208、及び214は単一のアドレス・ラインを示す。ANDゲート208には、探索信号178(図5)、及びマッチ信号168(図5)の否定値(NOMATCH)も送られる。探索信号も否定がとられ、「修飾された」書き込み信号と共にANDゲート214に送られる。ORゲート212は、2つのANDゲートから出力を受け取り、ワード選択信号(WORDN)を生成する。マルチプレクサ、又は他の論理機構の組み合わせによって同等の機能を提供することができる。

【0067】更新回路は、データ圧縮動作の間、データ探索が実行される時に活動化される。データ圧縮の間に探索が失敗した場合、その文字列はメモリ内の次に利用可能なアドレスに置かれなければならない。データ探索後にデータ・ワードをメモリに書き込むのに必要なクロック・サイクルの追加を排除するため、マッチが発生しなかった場合、ゲート208が高レベルになる。文字列が既にデータ・バス190上にあり、かつ利用可能な次のアドレスが既にアドレス・ジェネレータ170によってセットされているので、マッチの発生が示された直後に書き込みを実行することができる。従って、マッチ信号の否定(NOMATCH)は、利用可能な次のメモリ位置に関連するワード・ライン(WORDN)を活動化するゲート208を活動化する。

【0068】探索動作の間にマッチが見つかった場合、ワード選択ラインが使用禁止にされ、書き込み動作は行われない。例えば、外部マイクロプロセッサ書き込み動作の間では、メモリ内でマッチが発生しなかった時でも、修飾された書き込み信号を使用してデータ書き込みが強制的に行われる。この更新機能は、辞書への書き込みが「透過的」であり、余分なメモリ・アクセスを必要としないので、バイトあたり丁度1サイクルの性能を提供する。

【0069】代替案として、図6の回路を使用してメモリ内にデータ有効(data_valid)フィールドをセットすることもできる。例えば、図5のシステムは、メモリ内のマッチを検査する前に、新しい各文字列をメモリにコピーすることができる。マッチが発生した場合、ワード選択信号を使用して、新たに記憶されたデータ列に関連するデータ有効フィールドが活動化される。

【0070】<u>データ圧縮</u>

圧縮に関する回路144の動作において、マイクロプロセッサ(図示せず)はシステムを圧縮用に初期設定し、メモリ188をリセットする。マイクロプロセッサ制御信号(探索信号178、読み取り/書き込み信号164、リセット信号162)は、制御論理機構146(図4)を介して圧縮前データ・

インタフェイス152から得られる。リセット・ラインは、様々な初期設定動作に使用することができる。例えば、このリセット・ラインは、各メモリ位置に関連するデータ有効フィールドをリセットするようにメモリ188に接続される。更に、リセット・ラインは、アドレス・ジェネレータを、文字列を記憶するための開始メモリ位置に初期設定する。

【0071】単一入力文字を初期設定するために、いくつかの異なる技法を使用することができる。例えば、単一入力文字は、圧縮済みデータ・ストリームの一部としてアルゴリズムを用いて符号化することができる。又、それぞれ任意の単一入力データ文字を表す、符号化済みの値の集合をメモリにロードすることもできる。

【0072】読み取り/書き込み・ライン164は、アドレス・ジェネレータ170によって提供されたアドレスをアドレス・デコーダ184に接続するようマルチプレクサ174に命令する。圧縮前データ・インタフェイス138(図4)からの外部文字列は、バス190のバイト・フィールド(DATA_IN[7:0])、及びコードワード・フィールド(DATA_IN[19:8])に供給される。次に、探索信号178が活動化され、それによってメモリ188は、コードワード/バイト列をメモリ188内の各位置と比較する。以前にメモリ188内には何も書き込まれていないので、最初はマッチは発生しない。従って、バス190上のコードワード/バイト列は、メモリ188内で最初に使用可能なアドレス位置(例えば、アドレス・ジェネレータ170によって生成される初期設定されたアドレス)に書き込まれる。次に、アドレス・ジェネレータ170がインクリメントされ、バス180からの新しい入力文字がメモリ・データ入力のバイト・フィールドに読み込まれる。この処理が繰り返され、マッチしないコードワード/バイト列のメモリ188への書き込みが続けられる。

【0073】マッチが成功した時、入力データ選択論理機構182は、エンコーダ194から生成されたコードワードをデータ・バス190のコードワード・フィールド(DATA_IN[19:8])に置くようマルチプレクサ192に命令する。次に、バス180からの新しい外部文字が、データ・バス190のバイト・フィールド(DATA_IN[7:0])に送られる。それによって、そのコードワードは、以前にマッチした文字列を表す。文字列に割り当てられたコードワードが、マッチしたデータ項目アドレスから直接導かれるので、入力文字を符号化するのに必要な制御論理機構は極めて小規模なものになる。更に、そのコードワードをマルチプレクサ192(MUX3)に送り返して次の入力文字と組み合わせることによって、各クロック・サイクルごとに1つの入力文字を処理することができる。

【0074】次に、新しいコードワード/バイト列がメモリ188内のデータ項目と比較される。この処理は、マッチが見つからなくなるまで繰り返される。この時点で、コンプレッサは、最後のマッチからコードワードを出力し、その新しいコードワード/バイト列をメモリ188に書き込む。次に、空(null)のコードワードと対になったバイトKをメモリで探索し、それによって新しい列を始めるためのルート・コードワードを生成することにより、バイト・フィールドに送られる最後の入力文字(K)が、更新済み辞書と比較される(「ルート」コードワードを含むように初期設定された辞書の場合)。次に、バス180からの新しい外部文字(K)がバイト・フィールドに送られ、マッチ処理が繰り返されて(LZWによって)新しい列が作成される。又、(LZ2と同様に)OMEGAの次に最後の文字Kを出力する、又はKのコードワードとしてOMEGAに続いてKのアドレスを出力することができる。

【0075】辞書が満杯になると、アドレス・ジェネレータ170は、もはや文字列をメモリに書き込むことができないことを、圧縮システムの残りの部分(図4)に対して示すテーブル満杯信号196を活動化する。その後、あらゆる追加入力データが、メモリ188に記憶されている現行の項目に応じて圧縮される。

【0076】<u>データ伸長</u>

データ伸長の場合、回路144においてメモリ188をリセットし、入力データを圧縮するために回路を初期設定することによって動作が開始される。伸長は、リンク・リスト伸長トラバーサルを含む。例えば、圧縮済みデータ・アドレスは単に、伸長済みデータ列が位置するメモリ内のアドレス(例えば、リンク・リストの「ルート」コードワード)を指すことができる。しかし、このアドレスは、「非ルート」コードワードを有することができる(例えば、このコードワードは、その符号化済み文字列を更に伸長するために必要な次のアドレスへのリンクである)。上述のように、「ルート」コードワード、及び「非ルート」コードワードは様々な方法で判定することができる。例えば、コードワードの値、又はメモリ内の識別ビットを用いることによって判定することができる。

【0077】圧縮済みデータ・インタフェイス148(図4)が利用可能な圧縮済みデータを有する時、そのデータは外部アドレス・バス177上でデコーダ184に書き込まれる。「非ルート」コードワードを受け取った後、メモリが読み取られ、(有効な位置を仮定して)バス186のバイト・フィールド(DATA_OUT[7:0])が制御論理機構146(図4)内のLIFOスタックにプッシュされる。バス186のコードワード・フィールド(DATA_OUT[19:8])は、非ルート・コードワードである場合、MUX1、及びMUX2を介してアドレス・デコーダ184に送り返され、他のメモリ読み取りが実行される。非ルート・コードワードがフィードバックされる前に、メモリから読み取られたデータ項目の最後のバイトがFIFO140内にプッシュされる。この処理は、メモリ読み取りの結果が「ルート」コードワードになった時に終了し、その時点で外部アドレス・バス177から新しいコードワードが読み取られる。

【0078】ルート・コードワードが識別された後、最後の復号化済み文字出力が、前の外部符号化済み文字と

連結され、メモリ188内の利用可能な次のアドレスに読み込まれる。読み取り選択論理機構172は、「ルート」コードワードに関する検査を行い、それに応じて、外部アドレス・バス177、又はDATA_OUTバス186をアドレス・デコーダ184に接続し直すようマルチプレクサ176に命令する。又、読み取り選択回路172は、完全に伸長されたコードワードを示すために、符号化済み要素個号198を制御論理機構146に供給する。FIFO140は次に、伸長された復号化済み文字をバス154上にダンプする。

【００７９】図５のシステムでは、伸長動作を簡略化している。伸長が、リンク・リスト・トラバーサルを含むので、組込み論理機構は、外部伸長論理機構（図４）への追加の相互作用なしにメモリ出力データをアドレス・デコーダに送り返すフィードバック機能を提供する。従って、各伸長サイクルに必要な時間が少なくなり、伸長制御論理機構が簡略化される。メモリ188内の有効なワード、及びコードワードを「修飾する」ための多数の異なる実施態様がある。ある方法では比較器技法を使用し、別の方法では、各ワードごとに余分のリセット可能なビットを使用する。ここで使用される技法は、特定のアプリケーション要件に依存する。単方向システム（例えば、CDROM）では、従来型のRAMを上述のフィードバック回路と共に、リンク・リスト・トラバーサルに使用して、伸長回路を更に簡略化することができる。

【００８０】図７は、本発明の上述の態様によるシステムにおけるデータ圧縮/伸長、又はリンク・リスト記憶/検索の一般的方法を示すデータ・フロー図である。以下に示す方法は、辞書がコードワード内に埋め込まれ、それによって、辞書が圧縮済みデータと共にそれぞれ転送されることが必要でなくなるようなシステムに適応させることができる。例えば、辞書が圧縮済みデータと共に転送される代替方法を、本システムを使用して実施することもできる。

【００８１】点線ブロック232はこのシステムの圧縮処理であり、点線ブロック234はこのシステムの伸長処理である。入力224における圧縮前データ(K)は、ブロック228から出力される符号化済み文字列(OMEGA)と共に決定ブロック226に供給される。上述のように、OMEGAは文字列を符号化するデータ項目のアドレスを表す。OMEGA、及びKは連結され、決定ブロック226で、辞書内の項目と比較される。OMEGA, Kの入力がメモリ内の項目にマッチする場合、ブロック228で、マッチしたデータ項目のアドレスを用いてその入力が符号化される。次に、この符号化済みの値（新しいOMEGA）が送り返され、次の外部文字Kと連結され、決定ブロック226に入力される。この処理は、OMEGA, Kの列がメモリ内のどの項目にもマッチしなくなるまで繰り返される。次にブロック230で、列テーブル・メモリがOMEGA, Kの列を用いて更新され、OMEGAが出力され、符号化ブロック228に文字Kが送られる。Kはブロック228で符号化され、決定ブロック226に送り返される前に、次の外部データ文字Kと連結される。

【００８２】符号化済みデータOMEGAは、伸長のためブロック236に送られる。所与の符号化済み入力文字(OMEGA(j))が、列テーブル・メモリにアクセスするためのアドレスとして使用される。決定ブロック238で、アドレスOMEGA(i)におけるデータ項目がルート文字であるかどうか判定される。ルート文字である場合、メモリから出力されたデータ項目内に追加する符号化済み文字はない（例えば、OMEGA(j)は存在しない）。次に、Kに関するメモリ・データ項目が伸長済み出力文字としてライン246上に出力される。決定ブロック238は、以前に符号化された文字(OMEGA(i-1))がKと連結され、利用可能な次のメモリ・アドレス位置に書き込まれた場合、ブロック240にジャンプする。次に、ブロック242は、入力ストリームにおいて次に符号化される文字(OMEGA(i+1))を、メモリから読み取られる次のデータ項目のアドレス位置として使用するようブロック236に命令する。

【００８３】列テーブル・メモリからの出力がルートでない（例えば、出力が符号化済み文字(OMEGA(j)と復号化済み文字K）から成る）場合、ライン246上にKが出力され、決定ブロック238はブロック244にジャンプする。ブロック244は復号化済み文字(OMEGA(j))を、メモリから出力される次のデータ項目のアドレスとして使用する。次に、メモリ位置OMEGA(j)におけるデータ項目が上述のように処理される。この処理は、全ての符号化済み入力文字が伸長されるまで繰り返される。

【００８４】図８は、図７における点線ブロック232の詳細データ・フロー図である。このデータ圧縮処理は、ブロック248で、開始信号、又はリセット信号が活動化された時点で開始する。ブロック250で、メモリ回路（以下で説明する）が、例えば、圧縮モード、又は伸長モードで動作し、辞書をリセットするように初期設定される。全ての辞書の有効ビットを初期設定する必要があり、好適には並行的に行われる。辞書は、項目を単一文字、又は「ルート」コードワードとして識別するために、単一文字コードワードを用いて初期設定することも、又は米国特許第4,558,302号で開示されたWelchのLZWやECMA-151規格で開示されたDCLZのような、選択された符号化アルゴリズムに従って外部で生成されたコードワードの集合を空(null)のコードワードと対にしたもので初期設定することもできる。又、コードワードの集合を事前に記憶するよりむしろ、例えば、本出願人に譲渡された、データ圧縮辞書アクセス最小化に関する米国特許第5,142,282号で開示されたように、マッチングが失敗するたびにコードワードをリアルタイムに生成することができる。空の辞書を含んだその他の初期設定技法を使用することもできる。

【００８５】入力データ・ストリーム内の最初の文字はブロック252で読み取られて、直接OMEGAフィールドに記

憶されるか、又は符号化され（例えば、コード(文字)）、次いでOMEGAフィールドに記憶される。次にブロック256で、入力データ・ストリーム内の次の入力文字(K)が読み取られる。ブロック258は、OMEGAとKを組み合わせて１つの文字列とし（即ち、OMEGA, Kの連結を作成し）、次にOMEGA, Kの列にマッチするデータ項目を辞書で探索する処理を示している。辞書にはまだデータ列が記憶されていないので、決定ブロック260はマッチがないことを示す。OMEGA, Kの列は現在、表現されていないので、決定ブロック266で利用可能な記憶空間があると判定された場合、メモリに記憶される。メモリが満杯でない場合、ブロック268の動作によって、利用可能な次のメモリ記憶位置(ADDR(N))にOMEGA, Kの列が自動的にロードされる。次にブロック270で、アドレス・カウンタをインクリメントし、メモリ内の利用可能な次の記憶位置(ADDR(N+1))が識別される。ブロック272で、第１入力文字の符号化済みの値OMEGA(アドレス)は適宜、符号化済みデータ列内の第１文字として出力される。

【００８６】メモリが満杯になると、圧縮システムは単純に、追加文字列をメモリに書き込む機能を停止する。例えば、決定ブロック266がメモリが満杯であると判定した場合、以下で詳細に説明するように、ブロック268の文字列をロードするステップ、及びブロック270のアドレス・カウンタをインクリメントするステップがスキップされ、この処理は、ブロック272の符号化、及び出力処理にジャンプする。

【００８７】OMEGAが出力された後、ブロック274のステップによって、第１入力文字(OMEGA)が第２入力文字(K)、又はコード(K)と置換される。次に、入力データ・ストリームからの次の入力文字が読み取られ(K)、それによって次のOMEGA, Kの列が提供される。次にこの処理は、メモリが新しいOMEGA, Kの列で探索されるブロック258にループ・バックする。

【００８８】決定ブロック260でマッチが示された場合、その処理はブロック264にジャンプし、そこでOMEGAフィールドが、マッチ・アドレスに等しいOMEGA, Kの列を表す符号化済みの値と置換される。次に、データ・ストリームからの次の入力文字がKフィールドにコピーされる。このOMEGAフィールドとKフィールドが組み合わされて、この場合３つの入力文字を表す新しいOMEGA, Kの列が形成される。この処理は、辞書のデータ項目が新しい文字列と比較されるブロック258に戻る。以前の文字列がメモリ内のデータ項目にマッチするかぎり、追加入力文字が文字列に追加される。もはや新しい文字列がデータ項目にマッチしなくなった時、決定ブロック260は、そこからブロック266にジャンプし、上述のようにブロック266, 268, 及び270のメモリ更新手順が実行される。ブロック272は、値OMEGA（例えば、最後の入力文字列/データ項目のマッチからの符号化済み文字列）を出力する。ブロック274は、この文字列内の最後の文字（例えば、その文字列が列テーブル内のどのデータ項目ともマッチしないような文字）をとり、OMEGAフィールドにコピーする。次にブロック274で、入力データ・ストリームからの次の入力文字がKフィールドにコピーされ、処理はブロック258にループ・バックする。それによって、圧縮処理から出力されたOMEGAの単一の符号化済みの値が複数の入力文字を表すので文字列が圧縮される。

【００８９】図９は、図７の圧縮回路246の詳細フロー図である。ブロック276で、列テーブル・メモリが伸長用に初期設定される。ブロック278で、第１の符号化済みワード(OLDCODE)が得られる。この入力読み取りステップ、又は以後の任意の入力読み取りステップの間で、これ以上のデータが利用できない場合、処理は終了する。ブロック280で、アルゴリズムに従うか、又は事前に列テーブル・メモリにロードされた項目を読み取るかのどちらかによって、第１の符号化済みワードが復号化される。第１の符号化済みワードはルート文字であり、従って復号化され出力される。

【００９０】ブロック282で次の符号化済みワード(INCODE)が得られ、ブロック284でINCODEが、列テーブルによって出力されるデータ項目のアドレスとして使用される。一実施例では最初、列テーブルは単一文字バイトのみから成り、従ってブロック284ではバイトKが出力される。次にブロック286でバイトKが出力される。以後のサイクルでは、ブロック284で以下に詳述するようにOMEGA, Kが返される。

【００９１】決定ブロック288で、そのバイトが列の終り（例えばルート文字）であるかどうかが判定され、そうである場合、ブロック292にジャンプする。ブロック292で、第１の符号化済み入力ワード(OLDCODE)と最後のバイト出力(K)を連結したものから成る新しいデータ項目が、列テーブル内の利用可能な次のアドレスに作成される。ブロック294で、次の未使用アドレス位置が指示され、ブロック296で、OLDCODEが最後の符号化済み入力ワード(INCODE)と置換され、ブロック282に戻る。

【００９２】ブロック282で、次の符号化済み入力ワード(INCODE)が読み取られ、ブロック284で、INCODEのアドレスにおけるデータ項目が出力される。INCODEのアドレスにおいて出力されたデータ項目がルートでない場合、それは復号化済みバイトKと、更に復号化できるように次のアドレスを指すコードワード・フィールド(OMEGA)を含んでいる。次にブロック286でKが出力され、決定ブロック288がブロック290にジャンプする。ブロック290でコードワード・フィールド(OMEGA)が、列テーブルから次に出力されるデータ項目のアドレスとして使用され、次にブロック284にループ・バックする。この処理は列テーブルから出力されるデータ項目がルート文字を含む（即ち、列の終りになる）まで繰り返される。次に決定ブロック288はブロック292に進み、そこで以前に読み取られた符号化済みワード(OLDCODE)が最後の出力バイト

(K)と連結される。次に、ブロック294、及び296の機能が実行され、処理はブロック282に戻る。従って、伸長処理によって、図８の圧縮処理で圧縮された最初のデータ・ストリームが再生成される。

【００９３】図10は、図８、及び９の圧縮、及び伸長アルゴリズムを視覚的に表した図である。生データ・ストリーム300は、図７に示したデータ圧縮/伸長処理に入力された圧縮前文字列から成る。この例では、初期設定時に、単一文字R、I、N、及びTがそれぞれ、メモリ302AのADDR0, ADDR1, ADDR2, ADDR3にロードされている。入力文字は各文字にそのアドレス位置の値を割り当てることによって符号化されるが、圧縮速度を高めるために、後述の処理を開始する前に単一入力文字をアルゴリズムに従って符号化することができる。メモリ302Aは、初期設定直後の状態の辞書を示し、メモリ302Bは、圧縮が完了した後の辞書を示す。

【００９４】データ・ストリーム300からの第１の入力文字Rは、アドレス位置ADDR0におけるデータ項目とマッチする。マッチがあったので、圧縮システムは、Rに関する符号化済みの値(ADDR0=0)を次の入力文字「I」と連結し、メモリ302Aで「0I」のマッチングに関して探索が行われる。メモリに「0I」のマッチングがないので、メモリ302Bに示すように、「0I」は利用可能な次のメモリ位置(ADDR4)に書き込まれる。マッチした最大のシーケンスに関するコードワード(即ち、「R」=0に関するコードワード)が、圧縮済み文字ストリーム304内で最初に符号化された文字として出力される。次に、圧縮システムは、次の入力文字「N」と連結された「I」に関する符号化済みの値(即ち、ADDR1=I)から成る列をメモリ302Bで探索する。302Bに示したように、列「IN」は辞書にないので、利用可能な次のメモリ位置(ADDR5)に書き込まれる。圧縮済み文字ストリーム304内で２番目に符号化された文字として値１(例えば、最後にマッチした文字列=「I」)が出力される。

【００９５】この処理は、圧縮前文字ストリーム300内の第２の「I」が処理される(例えば、文字306)まで、引き続き同様の方法でメモリ302Bを作成し、入力文字を符号化する。この圧縮システムは、「I」がアドレス位置ADDR1に位置するので、「I」を値１で符号化する。この符号化済みの値１が次の入力文字Nと連結され、その列「１Ｎ」がメモリ３０２Ｂ内のデータ項目と比較される。そのシーケンス「IN」は、文字ストリーム300に以前現れているので、列「IN」はメモリ内の項目(例えば、ADDR5におけるデータ項目)とマッチする。従って、列「IN」は「5」として符号化され、次の入力文字「T」と連結される。その列「5T」は、メモリ302B内のどの項目ともマッチしないので、利用可能な次のアドレス位置(ADDR8)に書き込まれ、最後にマッチした文字列「5」に関するコードワードが文字ストリーム304内に出力される。次に、入力文字「T」に関する符号化済みの値(ADDR3=3)が次の入力文字「I」と連結され、この処理が繰り返される。メモリ302Bは、文字ストリーム300から辞書のために作成された全ての文字を示す。文字ストリーム304は、生データ・ストリーム300に関する完全な圧縮済み文字ストリームである。文字ストリーム300内の９つの文字を表すのに６個の符号化済み文字しか必要とされないことに留意されたい。

【００９６】デコンプレッサ辞書は、302Cに示したように、最初の４つのアドレス位置がそれぞれ、単一入力文字R、I、N、及びTに関する復号化済みの値を含むように、伸長用に再び初期設定される。この場合も単一文字の復号化をアルゴリズムに従って実行することができる。第１の符号化済み入力文字「0」が、メモリ302Cへのアドレスとして使用される。伸長システムは、例えば値「0」が４より小さいことを検査することによって、その値がルート・コードワードであることを判定する。それによって、ADDR0におけるデータ項目(例えば、「R」が、伸長済み文字ストリーム308内の第１文字として出力される。次に、伸長システムは次の符号化済み入力文字「I」を読み取る。この値はやはりルート・コードワードであり、従ってADDR1におけるデータ項目が、伸長済み文字ストリーム308内の第２文字「I」として出力される。

【００９７】この時点で、以前のコードワード「0」と連結され、最後に伸長された文字「I」を使用して、新しい辞書項目が作成される。次に、メモリ302Dに示したように、その列「0I」が利用可能な次のアドレス位置(ADDR4)に書き込まれる。次のコードワード「2」が入力され、この処理が繰り返される。今回は、アドレス位置ADDR2におけるデータ項目(例えば、N)が出力され、次に列「IN」がメモリのアドレス位置ADDR5に書き込まれる。

【００９８】この処理は、入力文字「5」が伸長システムによって読み取られるまで、同様の方法で繰り返される。伸長エンジンは、このコードワードを使用してADDR5におけるデータ項目を参照する。符号化済み文字「5」は、３より大きいのでルートではなく、従って伸長システムは、アドレス位置ADDR5におけるデータ項目の最後のバイト(例えば、「N」)を出力する。データ項目の残り(例えば、「I」)は次のアドレスとして使用される。コードワード「I」はルートなので、ADDR1におけるデータ項目(例えば、「I」)が出力され、これ以上の伸長は必要とされない。次に、伸長済み文字「IN」が文字ストリーム308に置かれる。最後に伸長された出力文字「I」、及び以前に符号化された入力文字「3」を使用して、メモリ内の新しいデータ項目がアドレス位置ADDR7に書き込まれる。この処理は、文字ストリーム304内の文字が全て伸長されるまで繰り返される。HP-DC技法をハッシングと共に使用して作成された圧縮/伸長辞書は異なるものであることに留意されたい。これに対して、本システム、及び方法によって作成された圧縮辞書302Bと伸長辞書302Dは同一のアドレス/項目を有する。

【００９９】III. CAM圧縮/伸長システムにおける複数辞書の使用

CAMを使用してデータを圧縮するために必要とされるメモリ量を更に減らすために、図５に示したCAMデータ圧縮システムを待機辞書(図３参照)と共に使用する。CAMは、各クロック・サイクルごとに１文字を処理する能力があるが、現在、最小限のメモリを使用してデータを圧縮することができる。更に、リセット後、有用な文字列の集合を現行辞書に維持することによって、データ圧縮率が向上する。以下に示す方法は適応的なものであり、辞書は各伸長処理の前に個別の辞書が転送されなくても済むようにコードワードに埋め込まれる。

【０１００】図11は、CAM複数辞書の組み込まれた圧縮/伸長システムの高レベル・ブロック図である。例示のために、このシステムは、図５に示したものと同様の$2^b \times (b+m+2)$のCAM312を使用して実施されている。CAM312は、制御プロセッサ(図示せず)に接続された制御バス314を備えている。アドレス・バス316(ｂビット幅)、及びデータ・バス318(ｎビット幅)がCAM312に接続されている。ｎビット幅DATA_MASKバス320のゼロ・ビットはCAM探索時に、そのビットを使用不能にする。例えば、第１マスク・ビット(DATA_MASK[0])上の信号「0」は、CAM312に送られる第１のDATA_INビット(DATA_IN[0])を使用不能にする。使用不能とされたDATA_INビットは、ライン318上の信号にマッチするデータ項目をCAM312で探索する際に考慮されない。データ・マスキング回路は、当技術分野で周知のものである。従って、CAM312内で使用されるマスキング回路の細部は詳細には示されていない。ライン322のマッチング成功ラインは、バス318上のデータが、CAM312内に以前記憶された項目とマッチする場合には、必ず活動状態となる。MATCH_ADDRESSバス326は、マッチしたデータ項目のアドレスを含み、DATA_OUTライン324は、CAM312に以前記憶されたデータ項目を出力するために使用される。

【０１０１】図12は、CAM内の各辞書項目内に含まれる様々なフィールドを示す。各CAMデータ項目は、サフィックス文字Kを記憶するためのｍビット幅の文字フィールド(CHAR)、符号化済み文字の値OMEGAを記憶するためのｂビット幅のコード・フィールド(CODE)、及びコード・フィールドと文字フィールドに関連する辞書状況ビットを記憶するための２ビット幅の状況フィールド(ST)の３つのフィールドを有する。状況フィールド(ST)は、以下の４つのとりうる値のうち１つをとる。

FREE:CAMメモリ位置は現在、現行辞書で未使用である。

CD :CAM位置は、現行辞書に属するが待機辞書には属しないデータ項目を含む。

SD :CAM位置は、現行辞書と待機辞書の両方に属するデータ項目を含む。

INV :無効な値。通常の動作では発生しない。

【０１０２】FREE, CD, SD, 及びINVに対応する２進値は固定的ではない。コンプレッサ、及びデコンプレッサは、$0 \le S \le 3$のとりうる４つの状態(S)のどれか１つであってよい、状態マシンとして動作する。状況フィールド(ST)に関する特定の２進値は、状態(S)の関数FREE(S), CD(S), SD(S), 及びINV(S)であり、図13で定義されている。例えば、状態S=0ではCAMデータ項目の状況フィールドにビット[0:0]が存在する場合、そのメモリ位置はフリー(FREE)であり、現在、現行辞書で使用されていないとみなされる。しかし、コンプレッサ/デコンプレッサ・システムが状態S=2である場合、状況フィールドにビット値[0:0]を有するCAM位置は、待機辞書に割り当てられたデータ項目であるとみなされる。

【０１０３】最初、システムは状態S=0であり、全ての状況フィールドは[0:0]にセットされる(例えば、ST=FREE(S))。以下で詳細に説明するように、グローバル初期設定が必要とされるのはこの時だけであり、これによって、以後の辞書のリセット時に発生する恐れがある初期設定時間の遅延が最小限に抑えられる。最初は状態S=0であるコンプレッサは、入力文字の読み取り、入力文字の圧縮、及び現行辞書(CD)と待機辞書(SD)の作成を平行して開始する。CDが満杯になると、辞書の切り替えが行われ、それによってSD内のデータ項目が、CD内の新しいデータ項目になる。SDは、有効なデータ項目が全て削除され、基本的に空になる。

【０１０４】辞書の切り替えは、システムが状態をS=0->S=1に遷移させた時に行われる。図13を参照すると、状態S=1では、FREE項目は、状態S=0の場合のCDの値と同じST=[1:0]を有する。状態S=1では、CD項目は、状態S=0の場合のSDの値と同じST=[1:1]を有する。状態遷移が発生するのは、CDが満杯になった時だけであり、従ってCAM内の全ての項目はCD、又はSDのどちらかにマークされる(即ち、FREE値を有する状況フィールドの項目は存在せず、値INVが書き込まれることもない)。従って、S=0からS=1への状態遷移の直後に、全ての項目は値FREE、又はCDを有する(以下で説明するように、実際には辞書メモリ内に維持されない、最初の単一文字列は例外である)。値SDを有する項目はないので、新しいSDは空の状態で開始される。S=1->S=2, S=2->S=3, S=3->S=0の状態遷移に関しても同様の状況が発生する。図14は、上述の圧縮/伸長システムに関する状態遷移の変化を示す。

【０１０５】図15は、コンプレッサ/デコンプレッサの状態を変更するための簡単なハードウェア実施態様を示す。状況レジスタ328の最初のビット値が状態S=0において示されている。各状態遷移ごとに、状況レジスタ内のビットが、FREE->INV->SD->CD->FREEのように周期的にシフトしている。従って、状態制御は、８ビット周期のシフト・レジスタを使用して、各状態変化ごとにレジスタ328を２ビット左にシフトさせることによって簡単に実施される。

【０１０６】CAMベースの待機辞書コンプレッサを記述

する際、CAMメモリ位置の内容は３つのフィールド(状況,コード,文字)によって示され、コード(A)は単一文字列「A」の符号化済みの値を表す。説明上、コードワードには、メモリ・アドレス位置に対応する値が割り当てられている。しかし、コードワード値も、メモリ・アドレス位置の簡単な関数として容易に導かれ、当業者によって容易に実施される。事前に定義されたアドレス空間(例えば、アドレス０ないし2^n-1)内のコード(コード(A))は、辞書にアクセスすることなく、すぐに使用できると仮定する。前で説明したように(図５参照)、これらのコードに対応するメモリ位置はCAM内に物理的に存在する必要はない。従って、全てのCAM探索でこのような位置は除外されると仮定する。簡略化のために、「ファイルの終り」条件も無視する。

【０１０７】CAMベースの複数辞書システムの実施態様

図16は、CAMベースの複数辞書圧縮/伸長システムの詳細な回路図である。図16の回路図は、複数辞書圧縮/伸長を提供するために必要な、追加機能構成要素を示している。CAM圧縮/伸長回路312は、図11に示したものと同様であり、状況レジスタ328は図15に示したものと同様である。DATA_INレジスタ342、及びMASKレジスタ350はそれぞれ、CAM312のDATA_IN、及びMASKポートを介して、状況、コード、及び文字フィールドを供給する。CAM内の各データ項目に関する状況フィールドは、状況レジスタ328を介して直接制御されるか又は、状況パターン・ジェネレータ338を介して間接的に制御される。状況パターン・ジェネレータは図17に詳細に示されている。

【０１０８】DATA_INポートを供給している特定のCDライン、及びSDラインは、制御バス314を操作することによって、マルチプレクサ340(MUX M1)を介して制御される。制御バス314上の信号は、システム・プロセッサ(図示せず)、及びCAM312内の圧縮／伸長制御機能から得られる。制御バス３１４は、図５に示したのと同様の読み取り信号、書き込み信号、探索信号、及びリセット信号を含む。CAM312内のコンプレッサ/デコンプレッサ内部制御論理機構も図５に示したものと同様である。以下で記述する特定機能の幾つかを実施するために、この論理機構に対する軽微な修正が必要とされることがある。これらの回路の修正は、当業者によって容易に実施されるものであり、従って詳細には図示されていない。

【０１０９】ライン326は、CAM312のポートMATCH_ADDRESSをCAM312のポートDATA_INに接続する。外部データ・バス344は、直接ポートADDRESS_INに接続され、レジスタ342を介してポートDATA_INに接続されている。状況パターン・ジェネレータ・ライン336、及びデータ入力ライン348は、マルチプレクサ346(MUX M2)を介してMASKレジスタ350の状況フィールドを供給する。ライン349上の探索タイプの信号、及び制御生成回路352からのその他の様々な制御信号は、ライン322上のMATCH_SUCCESS信号によって制御される。ライン324上のDATA_OUT信号は、図４に示したように、圧縮済みデータ、又は伸長済みデータを、データ・インタフェイスに出力する。内部アドレス・ポインタ354(NEXT_CODE)は、第２アドレス・ポインタ356(SAVE_CODE)にデータを書き込むことも、あるいはCAMのポートDATA_INからデータを受け取ることもできる。

【０１１０】図17は、図16の状況パターン・ジェネレータ338の詳細な回路図である。状況レジスタ328(図16)のCDフィールド、及びSDフィールドからの第１ビットは、ANDゲート358、及びEXCLUSIVE-NORゲート362への入力である。CDフィールド、及びSDフィールドからの第２ビットは、ANDゲート360、及びEXCLUSIVE-NORゲート364に接続されている。ANDゲートはCAMのポートDATA_INの状況フィールドを供給し、EXCLUSIVE-NORゲートはCAMのポートMASKの状況フィールドを供給する(図16)。

【０１１１】圧縮/伸長システムは、(以下で詳述するように)CD辞書項目とSD辞書項目の両方を同時に探索できなければならない。これは、状況レジスタ328(図16)内のビットを操作することによって実行される。CD、及びSD内のビットの１つは常にマッチし、残りのビットは常に異なる(図13参照)。従って、そのマッチしたビットを使用して、有効なSD辞書項目、又はCD辞書項目が探索され、残りのビットがマスク・アウトされる。例えば、状態S=0では、現行辞書CDのビット値が[1:0]であり、待機辞書のビット値は[1:1]である。これによって、ANDゲート358、及びEXCLUSIVE-NORゲート362の出力が高レベルに遷移し、ANDゲート360、及びEXCLUSIVE-NORゲート364の出力が低レベルに遷移する。従って、第１ビット位置に「1」が配置されているCAM辞書内のあらゆる状況フィールド(例えば、CD(S)、又はSD(S))は、CD、又はSDの有効な辞書項目として識別される。

【０１１２】データ圧縮

図16のシステムは以下のような方法でデータを圧縮する。システムは、図15に示したように、ビット値を状況レジスタ328にロードすることによって状態S=0にセットされる。CAM辞書内の全ての状況フィールドはST=FREE(S)(即ち[0:0])にセットされ、アドレス・ポインタNEXT_CODEはCAMの利用可能な第１アドレスにセットされる。図５に示したCAMに関して前述したように、全てのCAMアドレスが文字列の記憶に使用可能な場合、データ圧縮の間に単一入力文字をアルゴリズムに従って符号化することができる。しかし、単一データ文字がCAMに記憶される場合、符号化文字列を書き込むために使用可能な第１アドレスは通常、最後の単一文字位置の次のアドレス位置になる。

【０１１３】必要ならば、第１の入力文字は、第１データ文字を入力データ・ライン344から読み取り、入力文字/データ項目のマッチングのためのアドレスをライン326で生成することによって符号化される。次に、符号化済みの第１文字(OMEGA)がレジスタ342において、入力データ・ライン344からの第２の入力文字(K)と連結され、OMEGA.K文字列が生成される。OMEGA.Kの列にマッチするデータ項目に関して、CAMで探索が実行される。同時に、状況フィールドが状況パターン・ジェネレータ338によって生成された値にマッチするCD値、又はSD値(例えば、既にCD項目、又はSD項目として記憶されているOMEGA.Kの列)について探索される。探索タイプ信号(SEARCH_TYPE)349のビットは全て、マッチを探索する際に値「1」をとり、それによってCAMマスクのコード・フィールド、及び文字フィールドが使用可能となる。図17に示したように、マルチプレクサ(MUX M1)、及びマルチプレクサ(MUX M2)はそれぞれ、状況パターン・ジェネレータ338からのポートMASK、及びDATA_INに関する状況フィールドを選択する。

【０１１４】これはCAMに供給された最初のOMEGA.Kの列なので、ライン322上のMATCH_SUCCESS信号はマッチがないことを示す。続いて、OMEGAがライン324上で出力され、文字列CD(S).OMEGA.Kがそれぞれ、CAM辞書位置NEXT_CODEにおける状況、コード、及び文字フィールドに書き込まれる。次に、OMEGA.Kの列の文字Kが符号化され(コード(K))、OMEGAに関する新しい値として使用される。状況フィールドに書き込まれたCD(S)値は、レジスタ342の状況フィールドに供給されるマルチプレクサ340の入力を修正することによって、直接レジスタ328から供給される。

【０１１５】システムは次に、利用可能な次のCAM辞書項目(例えば、ST=FREE(S))を探索する。従って、探索タイプ信号349は値「0」をとり、コード、及び文字フィールドをマスク・アウトし、ライン348上のビット値[1:1]を介して状況フィールドを使用可能にする。同時に、マルチプレクサ340に接続された制御ライン314は、レジスタ328からの値FREEを、状況フィールド内で探索された値として選択する。ライン326からのマッチ・アドレスは、次の固有なOMEGA.Kの列を記憶するためのNEXT_CODEとして使用される。この処理は、ライン344上の入力データ列から次の文字を抽出し、それをOMEGAと連結して、次の探索のためにOMEGA.Kの列を生成する。次の探索でマッチが見つかった場合、マッチのアドレス位置が、次のマッチを行うためにMATCH_ADDRESSライン326を介してポートDATA_INにフィードバックされる。このアドレスは、以前のOMEGA.Kの列を表す新しいOMEGAの値として使用される。同時に、レジスタ328からのSD(S)値がマッチ・アドレスにおける状況フィールドに書き込まれる。

【０１１６】上述のように、新しいOMEGA.Kの列がCAM位置に書き込まれた後、状況フィールド内の次のFREE値を見つけるための探索が実行される。探索の失敗は、現行辞書が満杯であることを示し、その場合、システムは状態をS=1に切り替える。これは、レジスタ328の内容を左に２ビット位置だけずらすことによって実行される。以前にSD(S)値を有していた状況フィールド位置は、今度はCD(S)を構成する。状態S=0の場合に、CAM内の状況フィールドは全てCD(S)、又はSD(S)のどちらかにセットされていた(例えば、状態が変化する直前にはFREE状況フィールド値は存在しない)ので、状態S=1でのフリー・メモリ位置は全て、状態S=0における以前のCD(S)項目である。更に、待機辞書は、INV値が状態S=0において書き込まれることがないため、状態S=1における最初の単一文字列を除き空になる。圧縮は、システムがS=1の状態で上述のように継続される。この処理は、入力データの全てが圧縮されるまで、引き続き圧縮済みデータ文字を生成し、状態を切り替える。

【０１１７】データ伸長

図16のシステムを使用するデータ伸長は以下のような方法で実行される。CAMは、レジスタ328内のビットを状態S=0にリセットすることによって初期設定される。利用可能な各メモリ辞書位置の状況フィールドにFREEビット値が書き込まれる。内部アドレス・ポインタ354(NEXT_CODE)が、CAM内の利用可能な第１のメモリ位置にセットされ(例えば、NEXT_CODE=2ₕ)、内部アドレス・ポインタ356(SAVE_CODE)がゼロにセットされる。

【０１１８】伸長は、図５で上述したのと同様の方法で実行される。例えば、符号化済み文字列の第１の符号化済み文字(OMEGA)がライン344上で読み取られる。次に、OMEGAがCAMのポートADDRESS_INに供給されるアドレスとして使用される。ライン324上で出力されるコード・フィールドの値が「ルート」でない場合、文字フィールドがライン324上で出力され、コード・フィールドが次のアドレス位置としてCAMにフィードバックされる。この処理は、「ルート」のコード・フィールドがCAMから読み取られるまで繰り返される。

【０１１９】圧縮済み入力文字(OMEGA)が伸長され、伸長済み文字列がライン324上で出力された後、アドレス位置OMEGAにおける状況フィールドがSD(S)にセットされる。これは、レジスタ328からのSD(S)値をレジスタ342の状況フィールドに書き込むことによって実行される。次に、伸長済みデータ列からアドレス位置SAVE_CODEにおけるレジスタ342の文字フィールドに第１の文字(K)をフィードバックすることによって、辞書が作成される。状況レジスタ328からのCD(S)値、最初にライン344を介して読み取られたOMEGA値、及び伸長済みOMEGA出力列からの第１文字(K)が、アドレス位置NEXT_CODEにおけるCAM辞書の状況、コード、及び文字フィールドに書き込まれる。次に、アドレス・ポインタNEXT_CODEの値がアドレス・ポインタSAVE_CODEに書き込まれる。値「0」がライン349上に配置され、ライン348上のビット値[1:1]によって「状況フィールドのみ」の探索が可能になる。次に、状況フィールドでFREE値を探索することによって、FREE状況フィールドを有するCAM内の次の辞書項目が見つけられる。FREE状況フィールドのアドレス値が、ライン326を介してアドレス・ポインタNEXT_CODEに書き込まれる。その後、次の符号化文字OMEGAがライン344から読み取られ

る。

【０１２０】現行辞書が満杯である（例えば、FREE状況フィールド値が存在しない）場合、上述のようにレジスタ328内のビットをシフトすることによってシステムが状態S=1に切り替えられ、アドレス・ポインタNEXT_CODEの値がリセットされる。従って、現行辞書は以前の待機辞書の項目しか含んでいない。システムはその後、次の符号化済み文字（OMEGA）をライン344から読み取り、そのデータを伸長する処理が継続される。

【０１２１】図18は、待機辞書を有するCAMを使用するデータ圧縮の一般的方法を示すデータ・フロー図である。ブロック376は、そのシステムのための状態条件、及び状況条件をセットする初期設定処理である。具体的には、システムが状態S=0にセットされ、CAM辞書内の状況レジスタが全てST=FREE（0）にセットされ、アドレス・ポインタが、CAM内の利用可能な次のアドレスに関してセットされる（例えば、2ⁿ->NEXT_CODE）。

【０１２２】ブロック378で、入力データ・ストリームの第１文字（CHAR->K）が読み取られ、値OMEGAを提供するように符号化される（例えば、コード（K））。ブロック380で、入力データ・ストリーム内の次の入力文字（K）が読み取られる。ブロック382で、OMEGAとKが文字列として組み合わせられる（例えば、OMEGA, Kとして連結される）。次に、OMEGA, Kの列にマッチするデータ項目を探すだけでなく、２つの状況レジスタ・パターンのどちらか１つ（ST=CD（S）、又はST=SD（S））とのマッチングも行う探索が行われる。この探索では、有効な文字列（例えば、OMEGA, K）を示す、現行辞書の値、又は待機辞書の値のどちらかの値がCAM内に書き込まれているので、現行辞書の値と待機辞書の値の両方を探さなければならない。例えば、状況レジスタ値ST=CD（S）は、現処理における状態の間に、関連するコード・フィールド、及び文字フィールドに以前、OMEGA, Kの文字列がロードされていたことを示す。状況レジスタ値ST=SD（S）は、関連するコード・フィールド、及び文字フィールドにOMEGA, Kの文字列がロードされ、前記フィールドが現在のプロセッサ状態において少なくとも一度、第２のOMEGA, Kの文字列にマッチしていたことを示す。従って、両方の状況レジスタ値（CD（S）、及びSD（S））は、上書きすべきでない有効なCAMデータ項目を示す。

【０１２３】まだCAM内にデータ列が記憶されていない場合、決定ブロック384で、マッチがないことが示される。ブロック388で、符号化済みの値OMEGAが、符号化済みデータ列内の第１文字として出力される。利用可能な第１のCAMアドレス位置（NEXT_CODE）にOMEGA, Kの列が書き込まれる。アドレス位置NEXT_CODEにおける状況フィールド（ST）に、CAM内の有効なデータ項目を示す値CD（S）が書き込まれる。次にブロック388で、OMEGAが第２の入力文字の符号化済みの値と置換される（コード（K）->OMEGA）。

【０１２４】ブロック390で、ST=FREE（S）を有する利用可能な次のアドレス位置がCAMで探索される。FREE（S）値を有する状況レジスタが見つからない場合、CAM内の現行辞書は満杯である。それによって、決定ブロック392で、CAMを次の状態S=S+1 mod 4に変更することによって、現行辞書（CD）を待機辞書（SD）と置換する。状態を変更する間、各状況レジスタの値が、上述のように再び割り当てられる（図13参照）。状況フィールド値は、FREE->INV->SD->CD->FREEのように再割り当てされる。この処理はブロック380に戻り、そこで次の入力文字（K）が読み取られる。次に、上述のようなマッチング処理が繰り返される。現行辞書が満杯でない場合、決定ブロック392はブロック394にジャンプする。ブロック394で、FREE状況レジスタ値を有するCAM内の次のアドレスが判定され、そのアドレスをNEXT_CODEに割り当てる（例えば、MATCH_ADDRESS->NEXT_CODE）。この処理はブロック380に戻り、そこで次の入力文字（K）が読み取られる。

【０１２５】決定ブロック384でマッチが示された場合、処理はブロック386にジャンプし、そこでOMEGAフィールドが、OMEGA, Kの列にマッチした辞書項目のアドレスと置換される。マッチ・アドレスにおける状況フィールド（ST）をSD（S）にセットすることによって、（マッチ・アドレスにおけるコード・フィールド、及び文字フィールドによって表される）マッチした文字列が、自動的に待機辞書に割り当てられる。次に、この処理はブロック380に戻り、データ・ストリームから次の入力文字（K）が読み取られる。マッチ・アドレス（OMEGA）とKは、連結されて、３つの入力文字を表す新しいOMEGA, Kの列を形成する。次にブロック382は、現行辞書、及び待機辞書で文字列のマッチを探索する。

【０１２６】待機辞書の実際の作成は、OMEGA, Kの文字列のマッチ・アドレスにおける状況フィールドが、SD（S）にセットされた時にブロック386で行われる。これは、この位置が既にSD（S）によってマークされている可能性があるので、「重複作業」であることが多い。しかし、状況フィールドを検査しないということは、ハードウェアによる実施をより簡単にするのに役立つ。

【０１２７】図19は、待機辞書を有するCAMを使用するデータ伸長の一般的方法を示すデータ・フロー図である。ブロック398で、システムを状態（S=0）に初期設定し、全ての利用可能な辞書項目に関する状況フィールドを値ST=FREE（0）に初期設定する。アドレス・ポインタNEXT_CODEが第１のフリー・アドレス位置（NEXT_CODE=2ⁿ+1）にセットされ、第２のアドレス・ポインタSAVE_CODEがゼロにセットされる。ブロック400で、圧縮済みデータ列（OMEGA）からの第１の符号化済み文字が読み取られる。

【０１２８】ブロック401では、図16で上述したように、OMEGAを伸長済み文字列Wに伸長する。例えば、OMEGAをアドレスとして使用することにより、メモリ位置OMEGAにおける文字フィールドがCAMによって出力される。

アドレスOMEGAからのコード・フィールドが「ルート」でない場合、それはCAMに供給される次のアドレスとして使用される。その後、次のアドレスに関する文字フィールドが、次の伸長済み文字Kとして出力される。アドレスOMEGAにおけるコード・フィールドが「ルート」である場合、アドレスOMEGAにおける文字フィールドが出力され、アドレスOMEGAにおけるコード・フィールド、及び文字フィールドが、待機辞書に割り当てられる（例えば、SD（S）->ST）。ブロック402で、文字列Wの第１文字がレジスタCに割り当てられる。

【０１２９】アドレス・ポインタSAVE_CODEがゼロでない場合、決定ブロック403がブロック404にジャンプし、文字列CD（S）、SAVE_CODE、Cをアドレス位置（NEXT_CODE）におけるCAM辞書に書き込むことによって辞書が作成される。SAVE_CODEがゼロに等しい場合、又はブロック404で文字列が書き込まれた後、ブロック405で、アドレス位置OMEGAにおける状況フィールドが待機辞書に割り当てられ（SD（S）->（OMEGA））、現在のSAVE_CODE値がOMEGAの値と置換される。ブロック406で、値ST=FREE（S）を有する次の状況フィールドが探索される。FREEの状況フィールドが見つかった場合、決定ブロック408はブロック410にジャンプし、そこでマッチ・アドレスがアドレス・ポインタNEXT_CODEに割り当てられる（例えば、MATCH_ADD->NEXT_CODE）。次に、この処理はブロック400に戻り、圧縮済みデータ・ストリームからの次の符号化済み文字（OMEGA）が読み取られて伸長される。

【０１３０】FREE（S）値を有する状況フィールドがない場合、決定ブロック408はブロック412にジャンプする。次に、この処理が次の状態に変更され、それによって現行辞書が待機辞書に切り替えられる（即ち、S=S+1 mod 4）。又、これによって、以前の状態からの現行辞書項目がフリー位置になる。ブロック413で、ST=FREE（S）を有する次のフリー位置が探索され、アドレス・ポインタSAVE_CODEの値がゼロにリセットされ、ブロック410にジャンプする。ブロック410で、ブロック413で見つけられたフリー位置のアドレス値がアドレス・ポインタNEXT_CODEに割り当てられる。次に、ブロック410は、ブロック400に戻り、そこで圧縮済みデータ・ストリームからのデータが全て伸長されるまで処理が継続される。

【０１３１】図20は、図18、及び19の圧縮、及び伸長アルゴリズムを視覚的に表した図である。生データ・ストリーム414は、図18に示したCAM圧縮処理に入力される圧縮前文字列から成る。この例では、初期設定の間に、単一文字R、I、N、及びTがそれぞれ、メモリ416の位置ADDR0、ADDR1、ADDR2、ADDR3にロードされる。単一文字入力は、各文字にそのアドレス位置の値を割り当てることによって符号化される。しかし、圧縮速度を高めるために、後述の処理を開始する前に単一入力文字をアルゴリズムに従って符号化しておくことができる。メモリ416は、初期設定直後の状態S=0の辞書を示し、メモリ418は、現行辞書を待機辞書と置換する（例えば、状態S=0から状態S=1に変更する）直前の状態S=0の辞書を示している。メモリ420は、生データ・ストリーム414を圧縮した後の状態S=2の辞書を示している。

【０１３２】辞書内の各メモリ位置は、状況フィールド（ST）、コード・フィールド（CODE）、及び文字フィールド（CHAR）に分割されている。図示を目的とするため、文字列を記憶するために使用可能なCAM内の辞書位置は５つしかないと仮定する（例えば、ADDR4ないしADDR8）。アドレス位置ADDR0ないしADDR3は、単一文字のために指定され、利用可能な辞書位置としては探索されない。各状況フィールドのビットは値FREE=[0:0]に初期設定され（例えば、FREE（S）=0）、アドレス・ポインタNEXT_CODEは、利用可能な第１のCAMメモリ位置に初期設定される（NEXT_CODE=ADDR4）。

【０１３３】生データ・ストリーム414からの第１の入力文字「R」は、アドレス位置ADDR0におけるデータ項目にマッチし、OMEGAの第１の値として使用される（例えば、OMEGA=0）。圧縮システムは、OMEGAを次の入力文字「I」と連結し（OMEGA, K）、メモリ416内のコード・フィールド、及び文字フィールド内で「0I」のマッチを探索する。同時に、メモリ416内の対応する状況フィールドで、ビット組み合わせ「1:0」、又は「1:1」（例えば、状態S=0におけるCD（S）、又はSD（S））が探索される。全てのメモリ位置はフリーであり、以前にメモリに書き込まれていた列「0I」はなく、従って、マッチは見つからない。従って、OMEGAの値（「0」）が圧縮済みストリーム422内の最初の文字として出力され、文字列（CD（S）, OMEGA, K）が第１のフリー・メモリ位置（ADDR4）に書き込まれる。次に、文字「I」が符号化され、OMEGAの次の値（例えば、OMEGA=1）が生成される。

【０１３４】FREE値を有する次の状況フィールドがCAM辞書で探索され、そのアドレス位置がアドレス・ポインタNEXT_CODEに割り当てられる（例えば、NEXT_CODE=5）。次に、生データ・ストリームの次の文字（K=「N」）がOMEGA（OMEGA=「1」）と連結され、CAMで文字列「1N」が探索される。この場合も、CAMでマッチは発生せず、文字列（CD（S）, 1, N）がアドレス位置ADDR5に書き込まれる。この処理は同様の方法で繰り返され、以前のどの項目にもマッチしない文字列が見つかった後、次の利用可能なアドレスの状況、コード、及び文字フィールドへの書き込みが行われる。

【０１３５】最初の文字列/データ項目のマッチは、生データ・ストリーム414からの文字424の組み合わせ（「IN」）から発生する。文字「IN」は、符号化されたOMEGA, Kの文字列（「1N」）を含む。アドレス位置ADDR5のコード・フィールド、及び文字フィールドはそれぞれ「1」、及び「N」であり、状況フィールドは以前ST=CD（S）にセットされていたので、文字列のマッチが発生する。そのマッチ・アドレスは新しいOMEGA値（OMEGA=5）として使用され、ADDR

5におけるデータ項目は待機辞書に割り当てられる(S=0の場合、ST=SD(S)=[1:1])。生データ・ストリーム414から次の文字「T」が読み取られ、OMEGAと連結される。ここで、この場合3つの文字を表す新しいOMEGA, Kの列(「5T」)が、前述のように探索される。値「5T」を有するOMEGA, Kの列はCAM内に存在しないので、次の利用可能なアドレス位置(ADDR8)に書き込まれる。符号化済み文字「5」が圧縮済み文字ストリーム422に出力され、「T」に関する符号化済みの値が次のOMEGA値として出力される(OMEGA=3)。

【0136】メモリ418は、文字列「5T」をアドレス位置ADDR8に書き込んだ直後のCAMの状況を示している。この処理は、次のFREE状況フィールドをメモリ418で探索する。ADDR8がCAM内の現行辞書における最後の利用可能な位置であると仮定すると、FREE状況フィールドは見つからない。このことは、現行辞書が満杯であり、従ってシステムが状態S=1に変更されたことを示す。状態S=1では、状況フィールド・ビット値[1:0]がフリー・メモリ位置を構成し、ビット値[1:1]が現行辞書項目を構成する(図13参照)。従って、アドレスADDR5における文字列を除き、状態S=1での現行辞書内の辞書位置は全て、文字列を記憶するために使用可能である。状態が変化すると、アドレス・ポインタNEXT_CODEが第1のフリー・メモリ位置にリセットされる(NEXT_CODE=4)。

【0137】状態S=1のメモリ420を参照すると、次の入力文字426(「I」)が生データの文字ストリーム414から抽出され、次のOMEGA, Kの探索のためにOMEGAと連結される(「3I」)。列「3I」はメモリ位置ADDR7に存在するが、その位置における状況フィールドは現在FREEである。従って、マッチは見つからず、圧縮済み文字ストリーム422内の文字438としてその符号化済みの値「3」が出力される。文字列(CD(S), 3, I)がメモリ位置ADDR4に書き込まれ、文字「I」が次のOMEGA値として符号化される(OMEGA=1)。次のFREE状況フィールドのアドレス位置がアドレス・ポインタに割り当てられる(NEXT_CODE=6)。アドレス位置ADDR5は、その状況フィールドが状態S=0から状態S=1に切り替わった後の現行辞書項目を示しているので、スキップされることに留意されたい。

【0138】生データ・ストリーム414からの次の入力文字428がOMEGAに連結され、新しい文字列が構成される(「IN」)。アドレス位置ADDR5でマッチが発生し、従ってOMEGAにマッチ・アドレスの値が割り当てられ、アドレス位置ADDR5における状況フィールドが待機辞書に割り当てられる。状態S=1の待機辞書に対するビット割り当ては[0:1]である(図13参照)。生データ・ストリーム414からの次の入力文字がOMEGAに連結され、探索処理が繰り返される。この処理は、生データ・ストリーム414からの文字が全て圧縮されるまで、現行辞書が「満杯になる」たび毎にシステムの状態を変更し続ける。

【0139】メモリ432は、伸長用の初期設定の直後に伸長の準備ができたメモリを示している。メモリ434は、状態S=0から状態S=1に変化する直前の状態S=0のシステムを示している。メモリ436は、圧縮済み文字ストリーム422を伸長した後の状態S=1のデータ項目を示している。メモリ432内の辞書は、最初の4つのアドレス位置がそれぞれ、単一入力文字R、I、N、及びTの復号化済みの値を含むように初期設定される。この場合も、単一文字の復号化はアルゴリズムに従って実行することができる。システムが状態S=0にセットされ、全ての辞書の状況レジスタがFREE(S)にセットされる。アドレス・ポインタNEXT_CODEが、最初に使用可能な辞書位置(ADDR4)にセットされ、アドレス・ポインタSAVE_CODEがゼロにセットされる。

【0140】伸長は、前述のように行われ、OMEGAがメモリ432へのアドレス・ポインタとして使用される。圧縮済み文字ストリーム422からの第1の入力コードはOMEGA値を構成する(OMEGA=0)。この伸長システムは、例えば、その値が4より小さいことを検査することによって、値「0」がルート・コードワードであると判定する。それによって、ADDR0におけるデータ項目(例えば、「R」)が伸長済み文字ストリーム430内の第1文字として出力される。次に、アドレス位置OMEGAにおける状況フィールドがSD(S)にセットされる。

【0141】伸長済みコードワードからの第1文字K(例えば、「R」)をアドレス位置SAVE_CODEの文字フィールドに書き戻すことによって、辞書が再作成される。この場合、「R」はADDR0の文字フィールドに再度書き込まれる。次に、文字列(CD(S), 0, R)がアドレス位置NEXT_CODE(例えば、ADDR4)に書き込まれ、SAVE_CODEがNEXT_CODEの値にセットされる(例えば、SAVE_CODE=4)。次に、アドレス・ポインタNEXT_CODEにメモリ434内の次のフリー・アドレスの値が割り当てられる(例えば、NEXT_CODE=5)。

【0142】文字「I」が、圧縮済み文字ストリーム422から読み取られ、OMEGAの次の値となる。OMEGAが伸長され、伸長済み文字ストリーム430内の次の文字として復号化済み文字「I」が出力される。アドレスADDR1の状況フィールドがSD(S)にセットされ(例えば、[1:1])、伸長済みOMEGA値からの第1文字(「I」)がアドレス位置SAVE_CODE(ADDR4)における文字フィールドに書き込まれる。次に、文字列(CD(S), 1, I)が、アドレスNEXT_CODE(例えば、ADDR5)におけるメモリ434の状況、コード、及び文字フィールドのそれぞれに書き込まれる。NEXT_CODEの値は、SAVE_CODEの新しい値として使用される。次のFREE状況レジスタが見つけられ、NEXT_CODEがそのアドレスにセットされる(NEXT_CODE=6)。

【0143】この処理は、圧縮済み文字ストリーム422からの符号化済み文字「2」、及び「3」に関しても同様の方法で継続される。圧縮済み文字ストリーム422の最初の「5」は第1の非ルート・コードワードであり、アドレスADDR5におけるデータ項目は文字列「IN」である。従って、ADDR5におけるコード・フィールド(「1」)は、CAMによって読み取られる次のメモリ位置としてフィードバックされる。次に、ADDR1における出力(「I」)が、以前の文字フィールド「N」と共に、CAMによって出力され、ADDR5における状況フィールドがSD(0)にセットされる。伸長済みコードワードの第1文字(「I」)がメモリ位置ADDR7の文字フィールドに書き込まれ(例えば、SAVE_CODE=7)、文字列(CD(S), 5, I)がCAM位置のNEXT_CODE(例えば、ADDR8)に書き込まれ、SAVE_CODEにNEXT_CODEの値がセットされる(例えば、SAVE_CODE=8)。メモリ434は、この文字列をメモリに書き込んだ直後の現行辞書の状況を示している。

【0144】次の探索は、FREE値を含む状況フィールドがないことを示す。従って、システムは状態S=1に切り替えられ、状況レジスタの値が、図13に示したように再割り当てされる。メモリ436を参照すると、アドレス・ポインタNEXT_CODEに第1のフリー・メモリ位置(ADDR4)が割り当てられている。アドレス位置ADDR0ないしADDR3と、ADDR5はこの場合、現行辞書内の項目であり、一方アドレス位置ADDR4と、ADDR6ないしADDR8は、状態S=1のフリー位置を構成する。圧縮済み文字ストリーム422からの文字438がOMEGAにセットされ(OMEGA=3)、新しい伸長状態S=1で伸長される。この復号化済み文字「T」が伸長済み文字ストリーム430に出力され、ADDR3における状況フィールドに、状態S=1に関する値SD(S)(例えば、[1:0])が割り当てられる。SAVE_CODEがADDR8を指し、従ってその文字「T」がメモリ436内のADDR8における文字フィールドに書き込まれる。文字列(CD(S), 3, T)がアドレス位置ADDR4に書き込まれ、SAVE_CODEに値NEXT_CODEが割り当てられる。次のフリー辞書位置はADDR6であり、従ってアドレス・ポインタNEXT_CODEに割り当てられる。この処理は、圧縮済み文字ストリーム422内の文字が全て伸長されるまで同様の方法で継続される。

【0145】従来のLZ2実施態様では、単一文字ストリームに$C_0, C_0+1, C_0+2, \ldots, C_0+(2^m-1)$の順序で順次コードが割り当てられる。ここで、$C_0$はある小さな定数である(例えば、$C_0=0$)。新しい複数文字列は、$C_0+2^m, C_0+(2^m+1), \ldots, 2^b-2, 2^b-1$の順序でコードが割り当てられ、以後の各文字列はCAM内の順次アドレス値を有する。従って、列へのコードの割り当ては、単にC_0+2^mに初期設定されたカウンタを保持し、新しい辞書列が作成されるたびにインクリメントさせることによって行われる。これによって、コンプレッサは、辞書のリセット後に長さm+1のコードを使用し、続いて辞書内の項目数が次の、2のべき乗の値に遷するたびに出力コードの長さを1ビットだけインクリメントして、可変長出力コードを使用することができる。従って、出力コードの長さは、(m+1)とbの間で変化する。ここで、2^bが辞書の最大サイズである。これは、コンプレッサが辞書アドレス・コードのより短い時に、より短い出力コードを使用するので、圧縮率にある程度の向上をもたらす。このデコンプレッサは、コンプレッサとロック-ステップで辞書を作成し、圧縮コードの予想される長さの記録を保持することができる。

【0146】図16に示した処理では、新しい文字列に関する符号化済みの値は第1のフリー辞書位置のアドレスである。辞書切り替えの直後において、CDは、必ずしも連続していないCAM内の位置を有する、以前の待機辞書からの文字列から成っている。これらの列は、切り替えの後で、それらの古いアドレスを保持し、それによってそれらのコードを保持する。従って、FREEの探索によって返されるアドレス(コード)は連続シーケンスを形成しない。又、辞書のリセットの直後に、範囲$0 \leq C \leq 2^b-1$の範囲にあるあらゆる符号化済み文字列Cは潜在的に使用可能である。

【0147】その結果、出力ストリームは固定長コードを使用しなければならない。しかし、このことが圧縮率に及ぼす悪影響は重大なものではない。CDが辞書のリセット後に、一部だけ満杯となった状態で開始されるので、たとえCD内のコードが再配列された場合でも、コードを表すのに必要なビット数は最大ビット長(b)とかけ離れたものにはならない。例えば、実験では、現行辞書は通常、1/4ないし1/2程満杯となった状態で開始されることが分かっている。これは、コードが連続的な順序で整列された場合でも、切り替え後にb-1ビットが必要となることを意味する。しかし、第1の現行/待機辞書(CD, SD)を作成している間に可変長コードを使用できるか、又は、各リセットの後で現行辞書を再配列することができる。

【0148】圧縮結果

ソース・コード、実行可能なオブジェクト・コード、ASCIIデータ・ファイル、テスト・ファイル、ビットマップ・イメージ・ファイルを含む様々なタイプのデータにCAM複数辞書システムの圧縮処理、及び伸長処理が適用された。可変長出力コードを使用して、従来のLZW技法で上記ファイルを圧縮した。圧縮の全体結果を図21に示す。ライン440は、CAM複数辞書システムの圧縮率を示したグラフであり、ライン442は、標準LZWアルゴリズムでの圧縮率を示す。ライン440、及び442は、bの関数、即ち出力コード内の最大ビットの関数(即ち、辞書サイズのlog2)として圧縮率(最初のファイル・サイズ/圧縮済みファイル・サイズ)をプロットしている。

【0149】CAM/待機辞書の利点を強調するために、そのプロットにおいて、12ビットLZWコンプレッサによって達成された圧縮率を点線444で描いてある。そうすると、同じ圧縮率を達成するCAM複数辞書の処理に関するbの値が見つけられる。図21に示したように、CAM複数辞書システムは、1/2ないし1/4の辞書項目数を用いて、標準LZWコンプレッサと同じ圧縮率を提供する(例えば、1ビット少ない=必要なメモリ空間が1/2)。この圧縮率

は、従来のLZWコンプレッサのデータ項目より１、又は２ビットだけ長いCAM辞書項目で達成される。

【０１５０】明確化のために、待機辞書技法の簡単な実施態様を示した。圧縮率を更に向上させるために多数の修正を実施することができる。例えば、図18、及び19に示した圧縮/伸長処理は、入力アルファベットの全ての単一文字列の集合で辞書が初期設定されると仮定している。又、前述のように空の初期設定、又は中間の初期設定を使用することもできる。中間の初期設定と待機辞書の組み合わせに基づく処理も、非常に小規模の辞書を使用して高い圧縮率をもたらすことができる。

【０１５１】このシステムを実施するための更なる方法は、現行辞書が満杯になった直後には辞書を切り替えない方法である。その代わり、現行辞書は凍結され、圧縮率に基づいて辞書の切り替えが行われる（即ち、圧縮率があるレベルを下回った場合）。現行辞書を凍結している間、待機辞書を凍結させることもでき、又は次の辞書の切り替えまでそれを作成し続けることもできる。

【０１５２】待機辞書方法に固有の別の修正は、図13に示した状況フィールド値INVを使用することである。現在、INVは辞書項目に関して使用されていない。INVを使用して、SD2と示されている第２レベルの待機辞書を定義することができる。既にSDのラベルが付いている項目は、２回以上参照されると、SD2に変更される（現在のINV値に対する新しい名前）。辞書の切り替え時には、CD項目がFREEに、SD項目がCD項目に、SD2項目がSD項目になる。第１の待機辞書（SD）はSD2内の文字列の集合から開始され、新しいSD2は最初から開始される。この修正は、当業者によって、図16に示したシステムで容易に実施される。

【０１５３】こうして、待機辞書を現行圧縮辞書と並行的に作成するLempel-Zivデータ圧縮アルゴリズムの変形例を示してきた。現行辞書が満杯になると、待機辞書がそれと置き換えられ、新しい待機辞書が開始される。この待機辞書は、メイン辞書の文字列の選択された部分集合を含み、それによって同じメモリ・バッファ上で両方の辞書を実施することができる。好適なシステム実施態様は、連想記憶メモリ・モジュールを使用する。処理時間、及び回路の複雑さを低減させるために、パワー・アップ後に辞書の初期設定を不要にする簡単な状態遷移技法に基づいて、辞書の切り替えを行う。従って、CAM複数辞書コンプレッサ/デコンプレッサ・システムは、メモリの一部しか使用せずに、従来のデータ圧縮実施態様に匹敵する圧縮率を達成し、制御回路の複雑さはほんのわずかに増加するだけである。

【０１５４】<u>IV. CAMベースの複数辞書システムにおけるデータ圧縮/伸長の選択的上書き方法</u>

ここでは、図14で示した圧縮/伸長システムを使用する第２のLempel-Ziv待機辞書（LZSD2）データ圧縮、及び伸長方法を説明する。LZSD2では、全ての辞書項目を常に、文字列のマッチングに使用することができる。利用可能なCAM記憶位置を符号化済みデータ項目で満杯にし、利用可能な各記憶位置を常に辞書に割り当てることによって、通常辞書の切り替え後に発生する圧縮性能の低下を排除する。従って、各辞書の切り替え後に、以前に記憶された多数のデータ項目の使用を不可にする複数辞書切り替え技法と比べて、全体的なシステム圧縮性能は向上する。

【０１５５】<u>LZSD2圧縮</u>

LZSD2圧縮には３つの辞書が使用される。現行辞書（CD）は、新しい列、及び待機辞書から移された列を保持する。待機辞書（SD）は、辞書探索によってマッチし、従って「良好な」列である列を保持する。データ項目を待機辞書に割り当てるための代替基準もある。FREE事前辞書（FREE/PD）は、現在データ項目を割り当てられていない記憶位置、及び現行辞書から移されたデータ項目を含む。又、（FREE/PD）辞書内のデータ項目は、新しい文字列で選択的に上書きされる。FREE/PD位置が満杯になると、CAMが状態を変更し、続いて実際に辞書の切り替えを行う。

【０１５６】辞書の切り替えによって、データ項目が新しい入力文字列で上書きされる優先順位が変化する。例えば、CD内のデータ項目がFREE/PDに移され、SD内のデータ項目はCDに移される。従って、辞書の切り替えの後、以前CDに割り当てられており、新しい文字列で上書きされていないデータ項目が、今回FREE/PDに移され、符号化済み文字列で上書きされる。

【０１５７】図22ないし26を参照すると、全ての辞書空間の完全で連続した使用は一般に、３つの辞書SD、CD、及びFREE/PDを同時に探索することによって行われる。図22に示したように、最初、CAM内の利用可能な各記憶位置におけるコード・フィールド、及び文字フィールドは全て、既知の値、通常空文字（null）にリセットされ、FREE/事前辞書（FREE/PD）に割り当てられる。利用可能な記憶位置とは、データ項目を記憶するために使用できるCAM内のアドレス位置を指す。辞書データ項目は、それまでに見つかったうちで最良の辞書マッチのアドレスであるPREVCODE Nと、入力データ・ストリームからの最新の文字であるCH Nを含む列から成る。

【０１５８】第１の文字列（PREVCODE1, CH1）は、FREE/PD辞書内の利用可能な第１のアドレス位置（ADDR0）に記憶され、現行辞書（CD）に割り当てられる。次の固有文字列（PREVCODE2, CH2）は、利用可能な次のFREE/PD記憶位置（ADDR1）に記憶され、やはりCDに割り当てられる。文字列（PREVCODE3, CH3）、及び（PREVCODE4, CH4）は、それぞれCAMにおける利用可能な次のアドレスADDR2、及びADDR3に記憶され、共にCDに割り当てられる。

【０１５９】図23を参照すると、新しい固有文字列が、利用可能なFREE/PD記憶位置に継続されて記憶され、CDに割り当てられている。この圧縮処理が、既にデータ項目としてCAMに記憶されており、かつ上書きされていない新しい文字列を受け取った場合、そのデータ項目は、再割り当てされるか、又は待機辞書（SD）に移される。例えば、データ項目（PREVCODE1, CH1）は以前に、アドレス位置ADDR0に記憶されていた。従って、新しい文字列が（CODE, CH）の値（PREVCODE1, CH1）を含む場合、ADDR0におけるデータ項目はSDに再割り当てされる。

【０１６０】図24は、各データ項目が新しい入力文字列にマッチしたため、待機辞書に割り当てられた、以前に記憶されたデータ項目（PREVCODE1, CH1）、（PREVCODE3, CH3）、及び（PREVCODE4, CH4）を示している。図24に示したように、CAMは、利用可能な各CAM記憶位置（例えば、ADDR7におけるFREE/PD位置）が、CD、又はSDのどちらかに割り当てられたデータ項目で満杯になるまで現在の状態に維持される。図24は、辞書切り替え前にデータ項目と置換される、アドレス位置ADDR7における利用可能な最後のFREE/PD位置を示している。

【０１６１】全てのFREE/PD位置にデータ項目が割り当てられた後、CAMは状態を変更し、辞書切り替えを行う。図25は、辞書切り替え直後の各データ項目の状況を示す。例えば、以前SDに割り当てられていたデータ項目は全て、CDに再割り当てされ（SD->CD）、以前CDに割り当てられていたデータ項目は全て、FREE/PDに再割り当てされる（CD->FREE/PD）。辞書切り替えの後、データ項目は全て辞書に割り当てられたままであることに留意することが重要である。切り替えの後、待機辞書の項目はなくなる。例えば、以前にCDのアドレス位置ADDR1、及びADDR4ないしADDR7に割り当てられていたデータ項目は、FREE/PDに再割り当てされる。従って、CAMリセットの後でも、データ項目は全て、文字の符号化に使用することができる。従って、以前に符号化された圧縮データが、CAMリセットの後に失われることはないので、データ圧縮性能は維持される。

【０１６２】LZSD2の方法も、FREE/PDに割り当てられたデータ項目を、以前CAM内に記憶されたことのない新しい文字列と選択的に置換することによって、新しい入力データに適応する機能を有する。具体的には、新しい文字列がFREE/PD、又は現行辞書（CD）内のどちらかのデータ項目にマッチした場合、その新しい文字列は待機辞書（SD）に再割り当てされる。例えば、図26では、次の入力文字列（PREVCODE1, CH1）は、アドレス位置ADDR0におけるCDのデータ項目にマッチする。従って、ADDR0におけるデータ項目はSDに再割り当てされる。更に、入力文字列（PREVCODE5, CH5）はアドレス位置ADDR4におけるデータ項目にマッチする。従って、ADDR4におけるデータ項目は、FREE/PDからSDに再割り当てされる（PREVCODE5, CH5, FREE/PD）->（PREVCODE5, CH5, SD）。

【０１６３】入力文字列がどんな既存のデータ項目ともマッチしない場合、新しい文字列がFREE/PD辞書位置に追加され、最初はCDに割り当てられる。例えば、その入力文字列（PREVCODE9, CH9）はCAM内のどんなデータ項目にもマッチしない。従って、（PREVCODE9, CH9）は、最下位アドレス（即ち、ADDR1）を有するFREE/PDのCAM記憶位置に書き込まれ、CDに割り当てられる（PREVCODE9, CH9, CD）。最下位FREE/PDアドレス位置以外の基準に基づいてデータ項目を割り当てることも可能である。以後のリセット、及び上書きの前に、同じ列（PREVCODE9, CH9）が発生した場合、この項目はSDに移される。

【０１６４】CD辞書内の全てのデータ項目と、まだ上書きされていないFREE/PD辞書内のデータ項目は依然として、常に文字列のマッチに使用されることが分かる。CAM状態の変更後でも、データ項目は全て辞書に割り当てられたままなので、圧縮情報が失われることはない。従って、データ圧縮率を一時的に低下させることなく、その辞書を連続的に更新することができる。

【０１６５】圧縮処理において新しい入力文字列が識別される時に、上書きされるFREE/PD内のデータ項目を選択するためのいくつかの方法がある。上述のように、FREE/PD、CD、又はSD内のどのデータ項目にもマッチしない新しい文字列は、最下位アドレスを有するFREE/PD内の記憶位置に上書きされる。辞書が、（ハッシング技法を使用するものと比べて）単にアドレスをインクリメントすることによって最初に作成されると、最下位FREE/PDアドレスが最も古い辞書項目になる。しかし、この状況があてはまるのは、全てのFREE/PD位置が一度上書きされるまでに過ぎない。又、FREE/PD辞書内の個別のデータ項目を、新しい入力文字列との置換のために確定的に選択することもできる。

【０１６６】例えば、データ項目は、以前のデータ項目がどれだけ長く辞書に存在していたかに応じて、FREE/PD内に書き込まれうる。この例では、CAM記憶位置に書き込まれた順序を識別するタグを各データ項目に割り当てることができる。次に、LZSD2探索処理は、使用頻度が最低（LRU）であることを示すタグ値を有するFREE/PDデータ項目を選択する。FREE/PD内の使用頻度が最低のデータ項目は、符号化済み文字列にマッチせずに最も長い期間CAM内に存在していたデータ項目である。

【０１６７】ある状況におけるLRUデータ項目は、新しい文字列にマッチしない確率が最も高い。従って、LRUデータ項目を上書きすると、既存のデータ項目が置換される時に発生する恐れがある圧縮情報の損失を最小限に抑えることができる可能性がある。タグを使用してLRUデータ項目を識別することは、Bunton、及びBorriello著「PRACTICAL DICTIONARY MANAGEMENT FOR HARDWARE DATA COMPRESSION」（Communications of the ACM、1992年１月、第35巻、第１号）に詳細に記載されている。

【０１６８】図27は、LZSD2データ圧縮の一般的方法を示すデータ・フロー図である。ブロック450では、図４に示した圧縮/伸長システムが、LZSD2圧縮のために初期設定される。ブロック452で、入力文字列からの入力文字

(CH)が一度に1文字ずつ読み取られる。決定ブロック456でファイルの終り(EOF)条件が識別された場合、決定ブロック454で、それがこの圧縮サイクルで初回かどうかが検査される。EOF条件がその圧縮サイクルで初めてであった場合、決定ブロック454はLZSD2圧縮処理を終了する。EOF条件の検出がなされた時に、それがこの圧縮サイクルで初めてでない場合、ブロック460が以前マッチしたシーケンスPREVCODEを出力し、ブロック464が符号化済み出力文字が適切にフォーマットされることを保証するために、更に一掃を行う。

【0169】再び決定ブロック456を参照すると、EOF条件が識別されない場合、ブロック458で、全ての3つの辞書(即ち、FREE/PD、CD、及びSD)で、拡張された列(PREVCODE, CH)が探索される。この列(PREVCODE, CH)が、以前に記憶されたデータ項目とマッチした場合、決定ブロック462はブロック472にジャンプする。(PREVCODE, CH)がまだSDにない場合、ブロック472で、マッチした(PREVCODE, CH)データ項目を有するCAM位置が待機辞書に再割り当てされる。このマッチしたデータ項目は、状況ビットを変更することによってSDに再割り当てされる。次に、列(PREVCODE, CH)は、マッチしたデータ項目のメモリ・アドレスを使用して符号化され、PREVCODEに割り当てられる。即ち、コード(PREVCODE, CH)->PREVCODEである。次に、ブロック472は、次の入力文字(CH)がPREVCODEの符号化済みの値と組み合わされるブロック452にジャンプする。このように、マッチした列内の各サブストリングごとに記憶されたコードに関する状況ビットが更新され、以後のリセットにおいて保持される。

【0170】決定ブロック462で、マッチの発生しない時点まで列が拡張されると、ブロック466で、PREVCODEが、見つかったうちで最良のマッチとして出力される。利用可能なFREE/PD位置がある場合、ブロック468で、利用可能な次のアドレスに(PREVCODE, CH)を書き込むことによって辞書が更新され(例えば、最下位アドレスを有するFREE/PD辞書)、(PREVCODE, CH)が現行辞書(CD)に割り当てられる。利用可能なFREE/PD位置がない場合、ブロック468で、状況ビットを変更することにより(即ち、CD->FREE/PD、SD->CD)、現行辞書をFREE/PD辞書に切り替え、待機辞書を現行辞書に切り替えることによって辞書が更新される。ブロック470で、CHをPREVCODEに割り当てることによって(CH->PREVCODE)、次の入力文字列が準備される。単一文字列から圧縮済みコードへの代替マッピングが行われる。次に、この圧縮処理はブロック452に戻り、次の入力文字(CH)を読み取り、PREVCODEと組み合わせる。

【0171】図28ないし30は、図27に示したLZSD2圧縮技法の詳細データ・フロー図である。以下の変数は、LZSD2圧縮、及び伸長を記述するために使用される。

CAM　連想記憶メモリ。各辞書項目は(|MAXBITSビット・コード・フィールド|、|8ビット文字フィールド|、|2ビット状況フィールド|)を含む。

CD　辞書項目が現行辞書にあることを示す2ビット状況値。

CH　最も新しい入力文字を含む8ビット変数。

CODE_SIZE　現在、各出力コードに関して使用されているビット数。最小値は9で、最大値は辞書サイズで決定されるMAXBITSである。2^(MAXBITS)>=(辞書項目の数)+(ルート・コードの数(通常256))+(制御コードの数)。

DEPTH　圧縮時はPREVCODEで、伸長時はINCODEで表される列内の文字数を意味する変数。この変数のサイズはMAXDEPTHによって決定される。

EOF　これがセットされた場合、データ・ストリームの終りに達したため入力ストリームからデータを読み取る試みが失敗したことを示すフラグ。

FIRST_CHAR　INCODEで表される列の最初の文字を意味する8ビット変数。

FOUND_CODE　入力データ列にマッチするデータ項目が辞書で探索され、マッチが見つかると、MAXBITSビット変数に、マッチが見つかったアドレスが割り当てられる。

FREE/PD　データ項目が事前辞書にあることを示す2ビット状況値。この値はその位置が上書きできることも示す。

GROW　各圧縮済みコードごとにあと1ビットだけ読み取ることを開始するようデコンプレッサに通知するCODE_SIZEビット制御コード。

INCODE　圧縮済みデータ・ストリームから読み取られるMAXBITSビット変数の値。

INVALID　これは、辞書項目でないあらゆるMAXBITSビット・コードである。即ち、INVALIDは制御コード、又はルート・コードを表すことができる。

LAST_CODE_BUILT　最後に作成されたコードのアドレスを意味するMAXBITSビット変数。

MATCH　この標識は、辞書の探索で首尾良くマッチが見つかった場合、真である。

MAXBITS　出力コード内のビットの最大数。

MAXDEPTH　コードが表すことのできる最大の列の長さ。

NEXTCODE　新しい文字列で上書きされるべき辞書項目のアドレスを意味するMAXBITSビット変数。

PREVCODE　圧縮の間に、それまでに見つかったうちで最良の辞書のマッチに関するアドレスを意味するMAXBITSビット変数。伸長の間は、PREVCODEが、前のサイクルの間INCODEが何であったかを表す。

PREVDEPTH　伸長の間のみのPREVCODEの列の長さ。

SD　辞書項目が待機辞書にあることを示す2ビット状況値。

STACK　列の反転のためにMAXDEPTH LIFO待ち行列によって使用される8ビット。

SWAP_FLAG　辞書切り替えが必要な時に真である標識フラグ。

TCODE　INCODEを復号化する際に、一時記憶位置として使用されるMAXBITSビット変数。

TDEPTH　スタックが空である間、そのスタックの深さの記録を保持するために使用される一時変数。

【0172】図28を参照すると、開始時に、ブロック474は、図4に示した圧縮/伸長回路を、公知で一貫性のある状態にする。各辞書項目は所定の値にセットされる。例えば、コード・フィールド、文字フィールド、及び状況フィールドは通常、それぞれ16進数(HEX)の000、00、及びFREE/PDにセットされる。従って、CAM内のあらゆる辞書位置は、2文字列(NULL NULL FREE/PD)を含む。圧縮率を更に向上させるために、各辞書項目を異なる値の列に初期設定することも可能である。例えば、入力文字ストリーム内で頻繁に発生する文字列の組み合わせを、圧縮処理が開始される前にFREE/PD辞書に書き込むことができる。

【0173】出力フォーマットの制御は、圧縮済みデータ・インタフェイス138、及び148(図4)によって実行され、ブロック474によって空の初期状態にリセットされる。CODE_SIZE変数/レジスタは通常9のような最小値にセットされ、LAST_CODE_BUILTレジスタはINVALIDにセットされ、DEPTHレジスタは0にセットされ、SWAP_FLAGはセットされない。

【0174】ブロック476で、8ビット文字が入力文字ストリームから読み取られ、変数CHに割り当てられる。決定ブロック478で、入力ストリームの終りに達したためにデータ読み取りが失敗したかどうかが判定される(即ち、EOFフラグ)。EOF条件が検出された場合、決定ブロック486は、圧縮処理を終了し、残りの全ての符号化済み情報が出力される。データ読み取り処理が成功した(即ち、EOF条件が検出されない)場合、決定ブロック478は、LZSD2圧縮処理に進む。

【0175】データ読み取りが失敗すると、決定ブロック478は決定ブロック486にジャンプし、そこでPREVCODEの列の長さが検査される。DEPTH=0の場合、PREVCODEの列の長さは0であり、出力することができない。EOFフラグも、入力データ・ストリームの終りに達したことを示したので、DEPTH=0は、出力すべきものが残されていないことを示し、従って圧縮が終わり、ブロック486が圧縮処理を終了させる。初期設定の後のDEPTH=0だけが、入力されたデータがないことを意味する。DEPTH>0の場合、決定ブロック486は決定ブロック492にジャンプする。

【0176】データ読み取りが成功した場合、決定ブロック484で、DEPTH変数/レジスタの値が検査される。DEPTH=0の場合、PREVCODEの列の長さは0であり、辞書探索で使用することはできない。従って、ブロック480で、その入力文字CHがPREVCODEに割り当てられる(PREVCODE=CH)。この場合、PREVCODEは1文字の文字列を表し、従ってDEPTHは1にセットされる。次に、ブロック480はブロック476にジャンプし、次の入力文字CHが読み取られる。

【0177】DEPTH>0の場合、PREVCODEは有効な文字列を有し、圧縮処理はブロック488に引き継がれる。ブロック488で、全ての3つの辞書(FREE/PD、CD、及びSD)で同時に、コード・フィールド内にPREVCODEを有し、文字フィールドにCHを有するデータ項目(PREVCODE, CH)が探索される。全ての3つの辞書の同時探索は、単にコード・フィールド、及び文字フィールドだけを探索し、状況フィールドの値を無視することによって実行される。

【0178】複数のアドレスが(PREVCODE, CH)の列にマッチする可能性があり、従って複数のマッチを1つに減らさなければならない。これは、最小の値を有するマッチ・アドレスを選択する優先順位エンコーダを使用することによって行われる。(PREVCODE, CH)にマッチする位置が見つかると、ブロック488で、MATCHフラグがセットされ、マッチ・アドレスがFOUNDCODEに割り当てられ、決定ブロック490に進む。

【0179】辞書内で(PREVCODE, CH)のマッチを見つけるだけでは、FOUNDCODEが受け入れ可能な出力コードであるかどうかを判定するのに十分ではない。決定ブロック490で、更に2つのテストにパスしなければならない。第1に、デコンプレッサは、そのコードがコンプレッサによって作成された直後である場合、FOUNDCODEを正確に復号化しない。従って、FOUNDCODEはLAST_CODE_BUILTと等しくない、そのため、最後に作成された辞書項目は出力コードとして使用できなくなる。

【0180】又、ある希なケースでは、辞書内の列がMAXDEPTHより長くなることもある。MAXDEPTHより長いコードが出力された場合、デコンプレッサ列反転レジスタ(図4参照)がオーバフローし、エラーが発生する。これを防ぐために、決定ブロック490でFOUNDCODEの列の長さも検査する。

【0181】辞書内で(PREVCODE, CH)の列がマッチし、LAST_CODE_BUILTがFOUNDCODEに等しくなく、DEPTH<MAXDEPTHである場合、決定ブロック490はブロック482にジャンプし、そこで(PREVCODE, CH)の文字列にマッチする辞書データ項目が待機辞書(SD)に再割り当てされる。このデータ項目は、アドレス位置FOUNDCODEにおける状況フィールドをSDにセットすることによって再割り当てされる。次にブロック482で、PREVCODEが、それまでに見つかったうちで最良のマッチ列に等しい値にセットされ、即ちFOUNDCODEにセットされ(PREVCODE=FOUNDCODE)、DEPTH変数がPREVCODEの新しい列の長さにインクリメントされる(即ち、DEPTHは1だけインクリメントされる)。次にブロック482はブロック476にジャンプし、次の入力文字(CH)が読み取られ、新しい列を形成する新しいPREVCODE値に追加される(PREVCODE, CH)。

【0182】再びブロック490を参照すると、(PREVCODE, CH)の列が辞書データ項目にマッチしない場合、又はブロック490で実行される他の2つのテストのどちらか

にパスしない場合、後述のように、ブロック496でPREVCODEが出力され、ブロック514で(PREVCODE, CH)の列が現行辞書に割り当てられる(即ち、PREVCODE, CH, CD)。

【0183】出力される前に、決定ブロック492(図29参照)で、PREVCODE内のビット数が検査される。PREVCODEが2CODE_SIZE以上である場合、現在のCODE_SIZEビット数(例えば、9)で表すことはできない。この場合、ブロック494で、将来出力されるコードが全て追加ビットで表されるようにCODE_SIZEが1だけ増やされる。ブロック494では、出力が可能になる前にフォーマッタ回路(図4)によってバイト内にパックされなければならない、CODE_SIZEビットを使用してGROW制御コードも出力される。GROWコードは、将来のコードが全て現在のコード・サイズより1ビット長いようなデコンプレッサ(図32参照)に対する信号である。PREVCODEを出力するために1ビットより多くのビットが更に必要とされることがある。従って、ブロック494は決定ブロック492にジャンプし、PREVCODEを実際に出力する前に他のGROWコードを送らなければならないかどうか検査する。次にブロック496で、CODE_SIZEビット数を使用してPREVCODEが出力される。このCODE_SIZE数は、PREVCODEを出力する前にPREVCODEをバイト内にパックするために、フォーマッタによって使用される。

【0184】決定ブロック498は、以前に決定ブロック478で実行されたEOF検査の続きである。決定ブロック478で検出されたEOF条件は、最後の最良のマッチ・コード(PREVCODE)を出力するために、ブロック492における圧縮の主フローに戻る可能性がある(決定ブロック486参照)。更に、最後のコード出力は、最後の出力バイトを完全に満たさない可能性もある。統計的には、このようなコードは8つの出力コードのうち1つだけである。従って、ブロック500で、最終バイトが出力される前に、必要であれば、残りのビットにビット0、又は1を詰め込む。この時点で、圧縮処理が終了する。

【0185】EOFフラグが検出されない場合、決定ブロック502で、SWAP_FLAGがセットされているかどうかが検査される。SWAP_FLAGがセットされている場合、FREE/PDをGDと置換し、CDをSDと置換することによって辞書が切り替えられる(SD->CD, CD->FREE/PD)。辞書を切り替えても、実際にCAM内のデータは変更されないが、状況フィールドが圧縮エンジン(図17参照)によってどのように解釈されるかは変更される。辞書切り替えの後で、以前SDを表していた状況レジスタ・コードは今度は、CDを表し、CDを表していた状況レジスタ・コードは今度、FREE/PDを表し、FREE/PDはINVになり、INVはSDになる。INVは、FREE/PDが切り替え前で常に空であり、それによってINVは切り替え後も空の状態を保持するため、空のままとなる。又、INVが切り替え前で空であったので(INVは常に空である)、待機辞書(SD)も切り替え後は空である。

【0186】決定ブロック502で、SWAP_FLAGがセットされていないと判定された場合、決定ブロック506(図30)で、PREVCODE(DEPTH)内の文字数が最大許容の列の長さ(MAXDEPTH)より小さいかどうかを確認するための検査が行われる。DEPTH=MAXDEPTHの場合、文字列(PREVCODE, CH)は長すぎて出力コードとして使用できず、従って現行辞書に追加されない。DEPTH<MAXDEPTHであり、利用可能な位置が存在する場合、文字列(PREVCODE, CH)は現行辞書に追加される。

【0187】第1ブロック508で、CAM状況フィールドを検査し、同時にコード(PREVCODE)、及び文字フィールド(CH)を無視することによって、利用可能なFREE/PD位置がCAMで探索される。従って、マッチは、コードや文字フィールドの値にかかわらず成功する。複数のFREE/PDデータ項目がある可能性があるので、優先順位エンコーダは最小値を有するマッチ・アドレスを選択する。

【0188】上述のように、どのFREE/PDデータ項目の使用頻度が最も低いかに応じて複数のマッチのうちから1つを選択することもできる。例えば、項目がCAM内に記憶された順序を示すタグを、各辞書項目に関連付けることができる。その場合、優先順位エンコーダは、複数のマッチのうちから、最も使用頻度の低いデータ項目を選択する。その最も使用頻度の低いデータ項目はおそらく、新しい符号化済み文字列にマッチする可能性が最も低い文字列である。従って、その最も使用頻度の低いデータ項目を置換すると、わずかな圧縮情報を失う影響を最小限に抑えられる。優先順位選択についての代替方法を実施することもできる。

【0189】FREE/PD状況フィールドが見つかった場合、決定ブロック510はブロック514にジャンプし、(PREVCODE, CH)がCAMの、そのマッチしたアドレス位置に追加される。ブロック514で、列(PREVCODE, CH)がCAMのアドレスNEXTCODEに書き込まれ、その列が現行辞書に割り当てられる。列(PREVCODE, CH)は、新しい入力文字列とのマッチが発生し、それによってSDに移されるまで、CDに存在し続ける。それ以外では、(PREVCODE, CH)は、辞書切り替えが実行され、次にFREE/PDに再割り当て、又は移されるまで、CDに存在し続ける。

【0190】FREE/PDの探索が失敗した場合、ブロック512で、辞書切り替えが必要であることを示すSWAP_FLAGがセットされる。FREE/PD状況フィールドが見つからないということは、(PREVCODE, CH)で表される列が辞書に入力されないことも意味する。伸長辞書との同期を維持するために、辞書切り替えは、次の圧縮サイクルまで延期される(決定ブロック502参照)。

【0191】例えば、LZSD2デコンプレッサ(図31、及び32ないし34を参照)は、コンプレッサとは異なる順序で状況フィールドの更新を実行し、FREE/PD状況フィールドの探索が失敗した直後に辞書切り替えを開始する。LZSD2デコンプレッサは、所与のコードワードに関する状況フィールド・ビットを更新し、次に以前のコードワードを辞書に書き込もうとする。従って、次のコードの状況フィールドが更新されるまで、LZSD2圧縮処理における辞書切り替えを延期することによって、辞書切り替えが行われた時に、コンプレッサ辞書とデコンプレッサ辞書を一致させることができる。

【0192】伸長サイクルの間にコードワードが作成されなかった場合、ブロック516で、LAST_CODE_BUILTに値INVALIDがセットされる。そこで、最後に作成されたコードワードを将来のマッチングで使用することができる。従って、最大の列の長さ(DEPTH=MAXDEPTH)を超えたため、又は辞書が満杯になったために文字列(PREVCODE, CH)が作成されなかった場合、最後に作成された辞書項目は最後の(PREVCODE, CH)のマッチ・アドレス(即ち、FOUNDCODE)を指さない。従って、ブロック516で、LAST_CODE_BUILTは、次の探索動作でFOUNDCODEにマッチできないアドレスINVALIDにセットされる。ブロック518で、PREVCODEがCHと置換される(PREVCODE=CH)。PREVCODEはこの場合、1つの文字列を表すので、DEPTHは1にセットされる。

【0193】次に、ブロック518はブロック476にジャンプし、入力データ・ストリームから次の文字が読み取られる。LZSD2圧縮は、入力文字ストリーム内の文字が全て符号化されるまで継続される。

【0194】LZSD2伸長

好適実施例では、LZSD2圧縮に使用されるのと同じ3つの辞書(CD, SD, FREE/PD)を使用して、LZSD2伸長技法も実施される。コンプレッサで、新しい文字を記憶するための位置(FREE/PD状況レジスタ割り当てを有するもの)を使い果たすと、デコンプレッサは、上記でLZSD2圧縮に関して論じたのと同様の方法で辞書を切り替える。例えば、辞書切り替えの後では、CDはFREE/PDに、SDはCDに、SDは空になる。データ伸長の場合、処理インタフェイス152(図4)は、圧縮/伸長エンジン142を介し、圧縮前データ・インタフェイス138内の列反転待ち行列に至るまで圧縮済みデータ・インタフェイス148からの圧縮済みデータの流れを制御する。

【0195】図31は、LZSD2伸長を示す概略流れ図である。ブロック520で、LZSD2伸長に関して図4で示した圧縮/伸長システムが初期設定される。次に、ブロック522で、圧縮済み入力データ・ストリームから符号化済み入力列(INCODE)が読み取られ、その符号化済み文字が一時変数/レジスタに記憶される。この入力データ・ストリームは、図27、及び図28ないし30で上述したLZSD2圧縮技法によって以前に符号化された入力文字列を表す。決定ブロック524で、符号化済み文字列の終りを示すEOF条件の検査が行われる。決定ブロック526で、入力文字INCODEが制御コードであるか、又は符号化済み文字列であるかが判定される。

【0196】入力コードINCODEが制御コード(即ち、データコンプレッサ用の制御命令を含む文字)である場合、ブロック534で、コードが評価され、必要な応答が実行される。決定ブロック526で、INCODEが符号化済みデータ(即ちLZSD2圧縮エンジンからの圧縮済みデータを識別する文字)であると判定された場合、ブロック528で、ルート・コード(符号化済みデータからの単一復号化済み文字)が復号化され、そのルート・コードが圧縮済みデータ・インタフェイス148(図4)に位置するLIFOレジスタ(スタック)上にプッシュされる。次に、そのスタックが空になるまでスタックから文字をポップさせることによって、文字が伸長済みデータとして出力される。

【0197】ブロック530で、入力文字(FIRST_CHAR)、及び以前の入力コード(PREV_CODE)を組み合わせることによって伸長辞書が更新される。次に、FREE/PD辞書で、利用可能な記憶位置が探索され、列(PREVCODE, FIRST_CHAR)が、利用可能な次のFREE/PDアドレスに記憶され、現行辞書に割り当てられる。利用可能なFREE/PD辞書位置がない場合、伸長辞書が切り替えられる(CD->FREE/PD, SD->CD)。次にブロック532で、次の符号化済み列のための伸長技法が準備され、ブロック522に戻って別の伸長サイクルが繰り返される。伸長では、圧縮済みデータが、LZSD2コンプレッサで符号化される前の元の状態に正確に復号化される。圧縮済みデータがロスレスであり、各符号化済み文字内に全ての圧縮情報を含むことに留意することが重要である。

【0198】図32ないし34は、図31に示したLZSD2伸長技法を更に説明する詳細フローチャートである。伸長の開始時に、CAMは、コンプレッサが圧縮済みデータを作成した時点において存在していたのと同じ初期状態を有しなければならない。従って、図32のブロック534で、各CAM項目は、最初にLZSD2コンプレッサでセットされたのと同じ初期値でセットされる。例えば、各コード・フィールド、文字フィールド、及び状況フィールドには通常、それぞれ、16進(HEX)の値000、00、及びFREE/PDが割り当てられる。デコンプレッサは、その初期設定されたデータ項目を(NULL, NULL, FREE/PD)として解釈する。上述のような初期設定の代替技法も可能であるが、圧縮と伸長の両方の場合において同じでなければならない。入力データ・ストリーム・アンフォーマッタ(即ち、図4の圧縮済みデータ・インタフェイス148)は空の初期状態にリセットされる。CODE_SIZEは最小値(9)にセットされ、PREVDEPTHは、最初に伸長処理のループを通過することを示す0にセットされる。

【0199】ブロック536で、圧縮済み入力データ・ストリームから単一バイトが読み取られ、バイトがCODE_SIZEビット・コード(即ち、9ビット・コード)にアンパックされる。CODE_SIZEビット・コードを満たすのに更にビットが必要とされる場合、ブロック536で、CODE_SIZEビットがアンパックされるまで、追加バイトが読み取られる。次に、アンパックされたビット・コードが変数/レジ

スタINCODEに割り当てられ、残りの全ビットは次のコードで使用される。

【０２００】ブロック536での入力コードの読み取りが、ファイルの終り(EOF)に遭したために失敗した場合、決定ブロック538は伸長処理を終了させる。入力コードの読み取りが成功した場合、決定ブロック540に伸長処理が引き継がれる。INCODEが、予約されたコードである場合、決定ブロック540は決定ブロック544、及び546にジャンプし、予約された特定のコードを識別し、適切な動作が行われる。具体的には、決定ブロック544で、INCODEがGROWコードかどうかの判定がなされる。GROWコードは、各圧縮済みコードをあと１ビットだけ読み取ることを開始するようデコンプレッサに通知する。圧縮処理で他の制御コードを使用した場合、その制御コードもこの時点で評価される。

【０２０１】CODE_SIZE≧MAXCODE_SIZEは、現在のコード・サイズが全ての辞書項目を表すのに十分なものであり、従ってエラーだけがこの制御コードを復号化させることを示す。従って、GROWコードが出現した時に現在のCODE_SIZEが既に最大値である場合、決定ブロック546はブロック550にジャンプし、エラー信号が生成される。CODE_SIZEが最大コード・サイズより小さい場合、ブロック548で、コード・サイズが１ビットだけ増やされる。その場合、将来のコードは全て、以前より１ビット長くなる。次に、ブロック548はブロック536にジャンプし、次の入力コード(INCODE)が読み取られる。

【０２０２】再びブロック540を参照すると、INCODEが制御コードでない場合、ブロック542(図33)で、デコンプレッサがINCODEを復号化できるようにセットアップされる。INCODEは少なくとも１つの文字列を表すので、DEPTHはこの場合１にセットされる。INCODEは後で伸長技法において必要とされるので、復号化の間は異なる変数/レジスタTCODEが(一時的なINCODEとして)使用される。従って、ブロック542で、TCODEはINCODEと等しい値にセットされる。

【０２０３】決定ブロック552(図33)で、TCODEが単一文字列を表す(例えば、256より小さい)か、又は複数文字列を表す(例えば、256以上)かが検査される。複数文字列の場合、ブロック554で、CAMアドレスにおける文字TCODEがスタックの１番上に置かれる。次に、CAMアドレスTCODEにおける状況ビットをSDにセットすることによって、TCODEが待機辞書に割り当てられる。DEPTHの値は１だけインクリメントされ、スタック上の現在の文字数+1に等しくなる。前記「+1」は、ブロック542でカウントされた列の第１文字である。次に、CAMアドレスTCODEにおけるコード・フィールドがTCODEに割り当てられる。TCODEはこの場合、まだ復号化されていない残りの列を表している。

【０２０４】DEPTHがMAXDEPTHより大きい場合、スタックはオーバフローする。従って、決定ブロック556で、コードワードで表される文字数が検査され、ブロック558で、エラー・フラグが生成され、DEPTHがMAXDEPTHより大きい場合、伸長処理を終了させる。例えば、デコンプレッサに入力されるデータがLZSD2コンプレッサによって作成されたものでない場合、エラーが発生することがある。DEPTHがMAXDEPTH以下である場合、決定ブロック556は決定ブロック552にジャンプする。伸長処理はブロック554を通るループを続け、単一文字CHがTCODEから集められ、アドレスTCODEにおけるコード・フィールドがTCODEに再割り当てされる。

【０２０５】TCODEが256より小さい場合、決定ブロック552はブロック553にジャンプする。その場合、TCODEは単一文字列(ルート・コード)を表す。必要条件ではないが、全ての単一文字列は、その文字に対応するASCIIコードと同じコードにマップされる。これによって、CAMのルックアップを行わずに、TCODEを列INCODEの第１文字として直接、スタックの１番上に置くことができる。スタック内の第１文字は後で使用されるので、TCODEは別の変数/レジスタFIRST_CHARに記憶される。FIRST_CHARが使用される前にTCODEは変化しないので、単にTCODEを使用することによって変数/レジスタFIRST_CHARを削除することができる。しかし、FIRST_CHARは、図32ないし34をより容易に理解させるために使用されている。

【０２０６】この場合、DEPTHは、スタック上に置かれた最後の文字がブロック553ではカウントされなかったが、ブロック542に戻るので、スタック上の文字数に等しい。スタックを空にする間に、DEPTHは使用され変更されるが、最初のDEPTHの値が後で使用され、従ってブロック553で、そのDEPTHの値も(一時的なDEPTHとして)変数/レジスタTDEPTHに割り当てられる。

【０２０７】図34を参照すると、TDEPTHが０より大きい場合、スタックは空ではなく、決定ブロック560はブロック562にジャンプし、単一文字がスタックからポップ・オフされて出力され、TDEPTHがデクリメントされる。文字は、TDEPTH=0になるまで、スタックからポップ・オフされ出力される。

【０２０８】スタックが空になると、デコンプレッサは、圧縮済み文字ストリームから新しい符号化済み文字を読み取る準備ができる。しかし、次の符号化済み文字が読み取られる前に、決定ブロック564でまず、PREVDEPTHの値が検査される。PREVDEPTHが０とMAXDEPTHの間にある場合、組み合わされた列PREVCODE(以前に読み取られた符号化済み文字)、及びFIRST_CHARが、利用可能な次のFREE/PD位置に記憶され、CDに割り当てられる(PREVCODE, FIRST_CHAR, CD)。PREVDEPTH=0の場合、デコンプレッサを通るのが初めてであり、従ってCAMに追加すべき新しい列はない。PREVDEPTHがMAXDEPTH以上である場合、新しい列(PREVCODE, FIRST_CODE)は長すぎてCAMに入力できない。いずれの場合も、決定ブロック564はブロック574にジャンプし、新しい文字列をCAMに追加するための後述のステップをスキップする。

【０２０９】PREVDEPTHが０とMAXDEPTHの間にある場合、ブロック566において、辞書状況フィールドでFREE/PDの値を探査することによって、利用可能な次のCAM記憶位置がCAMで探索される。コード・フィールド、又は文字フィールドが何を含んでいるかにかかわらず、マッチは成功する可能性があるので、コード・フィールド、及び文字フィールドは探索されない。複数のアドレスがFREE/PDの探索条件を満たす可能性があるので、複数のマッチを１つに減らさなければならない。従って、ブロック566で、圧縮に使用された優先順位エンコーダを使用して最小値を有するマッチ・アドレスが選択される。FREE/PD位置が見つかった場合、MATCHフラグがセットされ、マッチ・アドレスがNEXTCODEに割り当てられる。

【０２１０】FREE/PD位置の探索が成功した(即ち、MATCHフラグがセットされた)場合、ブロック572で、CAMのマッチした位置(NEXTCODE)に列(PREVCODE, FIRST_CHAR)が追加され、この列が現行辞書(CD)に割り当てられる。FREE/PD位置の探索が失敗した場合、ブロック568で、辞書が切り替えられる、即ち、(SD->CD, CD->FREE/PD, FREE/PD->INV)。FREE/PD位置が見つからないということは、(PREVCODE, FIRST_CHAR)で表される列がCAMに入力されないことを意味する。

【０２１１】これによって、前の入力列PREVCODEが文字FIRST_CHARによって拡張される。列(PREVCODE, FIRST_CHAR)は、同じ列(PREVCODE, FIRST_CHAR)が圧縮済みデータ・ストリームで再び出現するまで、CDに存在し続ける。その後、(PREVCODE, FIRST_CHAR)データ項目は、次の辞書切り替えが実行された時に、SDに移されるか、又はFREE/PDに移される。

【０２１２】ブロック574で、PREVCODEはINCODEの値と等しい値にセットされる。PREVCODEを伸長の次のパスで使用して、次の入力コードの第１文字を拡張文字として使用することによって、辞書内に置くための新しい文字列(PREVCODE, FIRST_CHAR)を作成することができる。PREVDEPTHは、PREVCODEの列の長さの記録を保持するために、DEPTHの値と等しい値にセットされ、最大の長さより長い列がCAMに追加されるのを防ぐ。次に、この処理はブロック536(図32)に戻る。

【０２１３】従って、LZSD2が、辞書切り替えの後の辞書内のデータ項目全てを維持することによって、ロスレスの圧縮/伸長システムの全体圧縮率をいかに向上させるかが示されている。即ち、全てのデータ項目が、現在のデータ圧縮性能を維持しながら、新しい入力文字列にマッチさせる機能を保持している。従って、辞書切り替えの直後に圧縮性能が低下することはない。LZSD2は又、辞書に割り当てられた優先順位の最も低いデータ項目を選択的に上書きすることによって、新しい入力データに適応する機能も有する。従って、圧縮性能は、入力データ・ストリームの現在の性向で最適化される。

【０２１４】好適実施例において本発明の原理を説明し図示してきたが、そのような原理から逸脱することなく本発明の構成、及び細部を修正可能なことが明らかであろう。特許請求の範囲に記載された意図、及び範囲に含まれる全ての修正、及び変形について請求を行う。

【０２１５】以下に本発明の実施態様を列挙する。

【０２１６】１．　メモリ・ベースの辞書を使用して、文字列から成るデータを圧縮、及び伸長するための方法において、複数の記憶位置を含むメモリ装置を提供する方法であって、各記憶位置が文字列に関するコードワードをデータ項目として記憶するための固有のアドレスを有する上記方法と、メモリ装置の複数の記憶位置内に少なくとも第１、及び第２の辞書を定義する方法と、文字列に固有に対応するコードワードを各データ項目として記憶する方法と、第１、及び第２の辞書の少なくとも１つに、記憶された各データ項目を割り当てる方法と、入力データ文字列を表すコードワード値を生成する方法であって、前記コードワードが、入力文字列の一部に対応し、前記辞書の１つに割り当てられ、以前に記憶されたメモリ内のコードワードに関連付けられた前記方法と、現在辞書の１つに割り当てられているデータ項目の１つを、新しい文字列に関連する新しいコードワードで選択的に上書きし、それによって、データ圧縮、及び伸長処理の間で、常に文字列に関するコードワードを生成するために、第１、及び第２の辞書内のデータ項目全て使用する方法を含むことを特徴とする方法。

【０２１７】２．　前記辞書の少なくとも１つに上書きの優先順位が割り当てられ、データ項目が、前記１つの辞書に割り当てられたデータ項目の前記上書き優先順位に応じて選択的に上書きされることを特徴とする項番１に記載の方法。

【０２１８】３．　前記メモリ装置が複数の状態を有し、前記辞書の各データ項目に対する割り当てが、前記メモリ装置の現在の状態に応じて決定されることを特徴とする項番１に記載の方法。

【０２１９】４．　各辞書に上書き優先順位が割り当てられ、前記メモリ装置の状態の変化によって、前記辞書に割り当てられたデータ項目を新しい文字列で上書きできるように、前記辞書の少なくとも１つの上書き優先順位が変更されることを特徴とする項番３に記載の方法。

【０２２０】５．　文字列に関するコードワードをメモリ装置内に記憶する方法が、文字列に関する新しいコードワードで上書きできる前記第１の辞書内の記憶位置を見つけるステップと、前記第１の辞書の利用可能な記憶位置に新しいコードワードを新しいデータ項目として記憶するステップと、前記の新しいデータ項目を前記第２の辞書に再割り当てするステップと、前記第１の辞書内の利用可能な記憶位置が全て上書きされた後に、前記第２の辞書内のデータ項目を全て前記第１の辞書に再割り当てするステップを含むことを特徴とする項番１に記載

の方法。

【0221】6. 複数の記憶位置を有する連想記憶メモリを提供する方法と、連想記憶メモリの複数の記憶位置内に第1、第2、及び第3の辞書を定義する方法と、それぞれがデータ文字列に対応する、固有のコードワードを前記記憶位置にデータ項目として記憶する方法と、前記第1、第2、及び第3の辞書の少なくとも1つに各データ項目を割り当てる方法と、データ文字列を表すコードワード値を生成する方法であって、前記コードワード値が、文字列に対応し、前記辞書の1つに割り当てられ、以前に記憶されたコードワードにメモリ内で関連付けられている前記方法と、現在、どの前記辞書にもデータ項目として記憶されていない、新しいコードワードと選択的に置換できるように、前記辞書の1つにおいて各データ項目を優先順位付けする方法と、現在、前記1つの辞書に割り当てられている、優先順位付けされたデータ項目を、現在前記メモリ装置に記憶されていない新しい文字列に対応する新しいコードワードで選択的に上書きし、同時に、各辞書のデータ項目を全て使用して、圧縮、及び伸長処理の間で、常にコードワード値を生成する方法を含むことを特徴とする項番5に記載の方法。

【0222】7. 連想記憶メモリが複数の状態を有し、各データ項目に対する辞書割り当てが、連想記憶メモリの状態に依存することを特徴とする項番6に記載の方法。

【0223】8. 以前にコードワード値を生成するために前記データ項目が使用された回数に従って、より高い優先順位が、そのデータ項目が新しい文字列と置換される可能性のより小さいことを示す辞書に、各データ項目が移されることを特徴とする項番6に記載の方法。

【0224】9. データ項目が、単一辞書からの選択的置換に対してのみ使用可能であることを特徴とする項番6に記載の方法。

【0225】

【発明の効果】本発明によって、辞書ベースの圧縮/伸長システムで、辞書リセット時の悪影響を最小にし、統計的特性が変化する入力データの圧縮に関し、最小量のメモリでデータ圧縮能力を最大にすることが可能となる。

【図面の簡単な説明】

【図1】本発明による現行、及び待機辞書を用いたデータ圧縮システムのデータ・フロー図である。

【図2】図1の待機辞書データ選択処理の一例を示す詳細データ・フロー図である。

【図3】本発明による現行、及び待機辞書を実施するデータ圧縮回路の一例のブロック図である。

【図4】本発明を実施するデータ圧縮/伸長システムを示す高レベル・ブロック図である。

【図5】図4のメモリ、及び制御論理機構の詳細ブロック図である。

【図6】図5のアドレス・デコーダ内のの自動更新回路の論理図である。

【図7】本発明による連想記憶メモリ(CAM)を使用するデータ圧縮/伸長方法の概略データ・フロー図である。

【図8】図7のデータ圧縮手順の詳細データ・フロー図である。

【図9】図7のデータ伸長手順の詳細データ・フロー図である。

【図10】図8、及び図9の圧縮、及び伸長手順を視覚的に表した図である。

【図11】本発明による複数辞書を用いた圧縮/伸長システムで使用できるように設計されたCAMを示すブロック図である。

【図12】図11に示したCAM内の各辞書の項目内の異なるフィールドを示す図である。

【図13】図11の状況フィールド内の各圧縮/伸長状態ごとの辞書値を示す図である。

【図14】CAM内の複数辞書を用いた圧縮/伸長システムに関する状態遷移の変化を示す図である。

【図15】コンプレッサ/デコンプレッサ状態を変更するための簡単なハードウェア実施態様を示す論理図である。

【図16】図11に示したCAM内の複数辞書を用いた圧縮/伸長システムに関する主構成要素の詳細回路図である。

【図17】状況パターン・ジェネレータの詳細回路図である。

【図18】待機辞書によってCAMを使用するデータ圧縮の一般的方法を示すデータ・フロー図である。

【図19】待機辞書によってCAMを使用するデータ伸長の一般的方法を示すデータ・フロー図である。

【図20】図18、及び図19の圧縮、及び伸長方法を視覚的に表した図である。

【図21】CAM内の複数辞書を用いたシステムの圧縮結果と標準LZW圧縮技法の圧縮結果を示すグラフである。

【図22】第2のLempel-Ziv待機辞書(LZSD2)圧縮方法を視覚的に表した図である。

【図23】第2のLempel-Ziv待機辞書(LZSD2)圧縮方法を視覚的に表した図である。

【図24】第2のLempel-Ziv待機辞書(LZSD2)圧縮方法を視覚的に表した図である。

【図25】第2のLempel-Ziv待機辞書(LZSD2)圧縮方法を視覚的に表した図である。

【図26】第2のLempel-Ziv待機辞書(LZSD2)圧縮方法を視覚的に表した図である。

【図27】LZSD2圧縮を実行する一般的方法を示すデータ・フロー図である。

【図28】図27に示した手順の詳細データ・フロー図である。

【図29】図27に示した手順の詳細データ・フロー図である。

【図30】図27に示した手順の詳細データ・フロー図である。

【図31】LZSD2伸長方法の一般的方法を示すデータ・フロー図である。

【図32】図31に示した手順の詳細データ・フロー図である。

【図33】図31に示した手順の詳細データ・フロー図である。

【図34】図31に示した手順の詳細データ・フロー図である。

【符号の説明】

50, 136 データ・コンプレッサ/デコンプレッサ集積回路(IC)

52, 142 データ圧縮/伸長エンジン

54 データ・コンプレッサ・インタフェイス回路

56, 58 マルチプレクサ

60 読み取り/書き込み信号

64 アドレス信号

68 アドレス・ジェネレータ回路

70 データ列マッチ信号

72 辞書リセット要求信号

76 切り替え制御装置分岐回路

84, 86 トランシーバ

88, 90 ランダム・アクセス・メモリ(RAM)

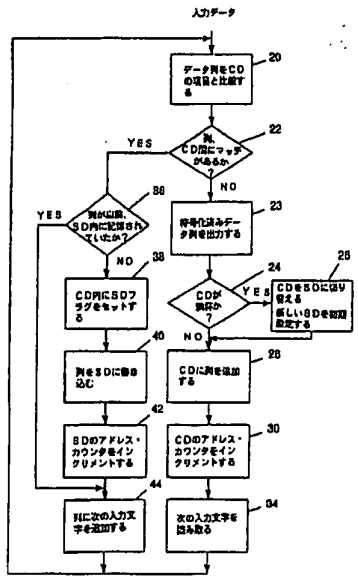
92, 96 待機状況フィールド

94, 98 データ項目フィールド

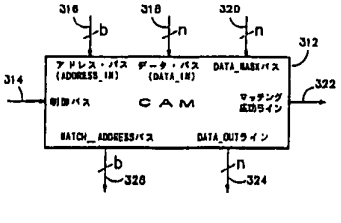
144 列テーブル・メモリ

146 制御論理機構

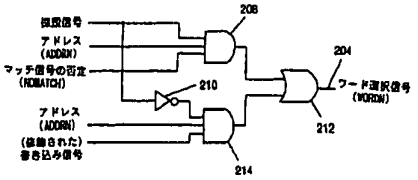
【図2】



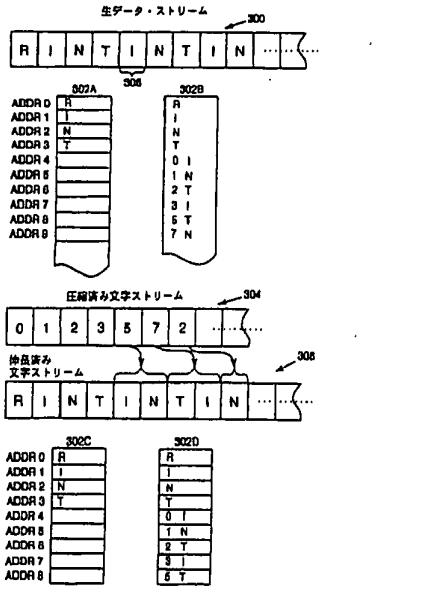
【図11】



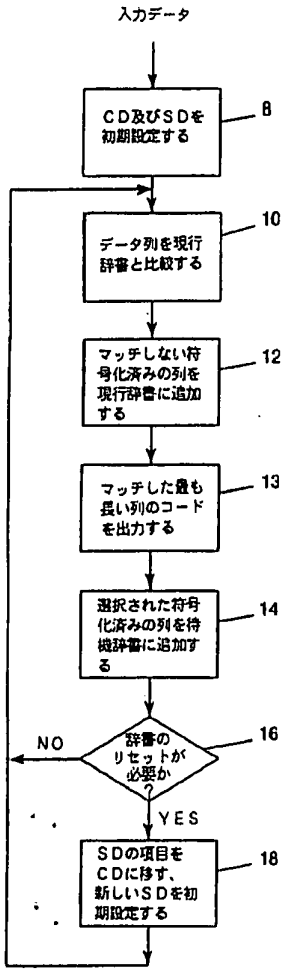
【図6】



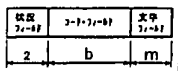
【図10】



【図１】



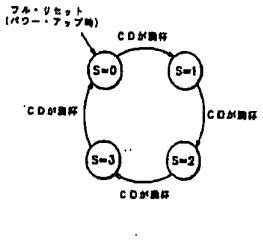
【図１２】



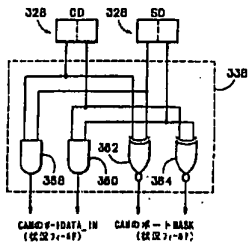
【図１３】

状況フィールド	状態 S=0	S=1	S=2	S=3
FREE	00	10	11	01
CD	10	11	01	00
SD	11	01	00	10
INV	01	00	10	11

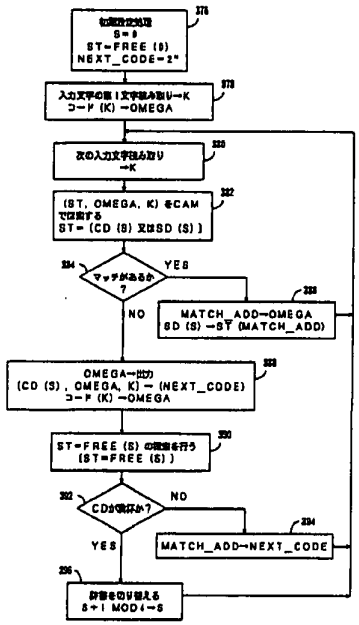
【図１４】



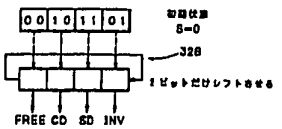
【図１７】



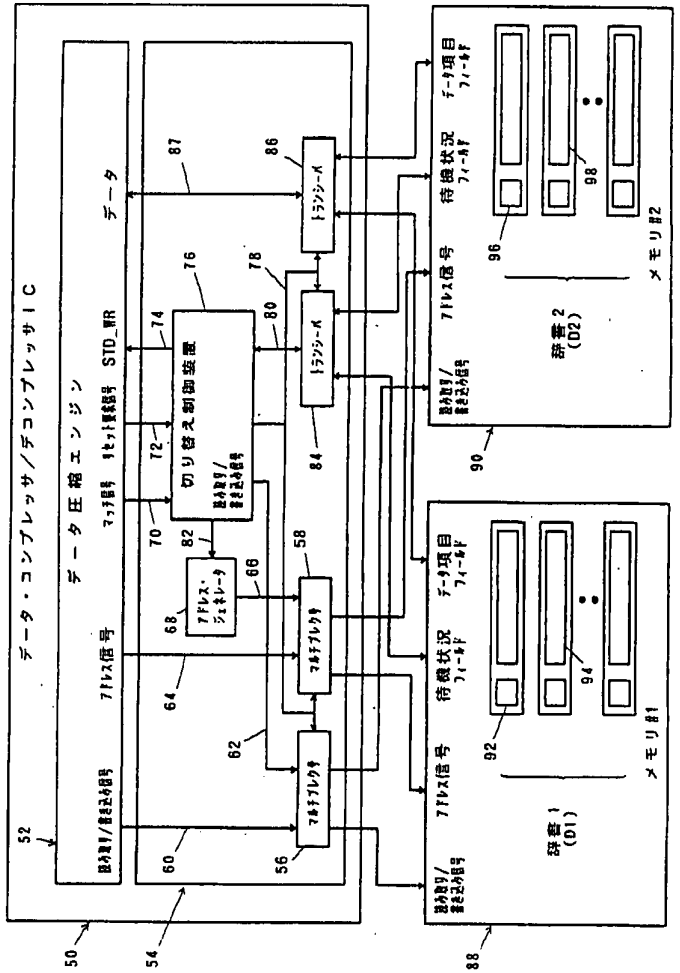
【図１８】



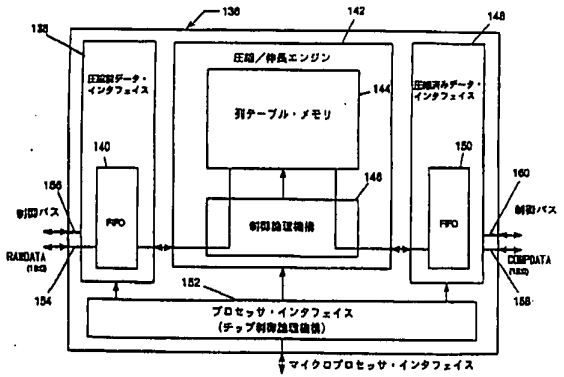
【図１５】



【図３】



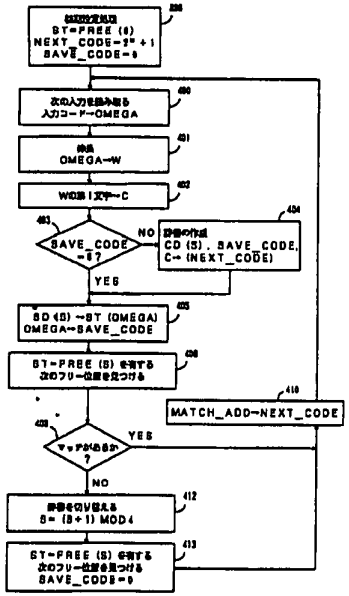
【図４】



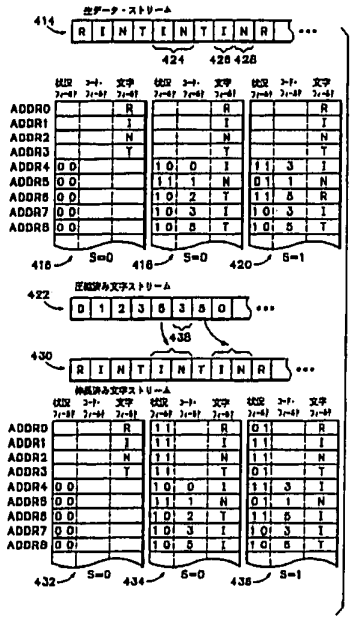
【図２５】

	コード・フィールド	文字フィールド	状況フィールド
ADDR 0	PREVCODE1	CH1	CD
ADDR 1	PREVCODE2	CH2	FREE/PD
ADDR 2	PREVCODE3	CH3	CD
ADDR 3	PREVCODE4	CH4	CD
ADDR 4	PREVCODE5	CH5	FREE/PD
ADDR 5	PREVCODE6	CH6	FREE/PD
ADDR 6	PREVCODE7	CH7	FREE/PD
ADDR 7	PREVCODE8	CH8	FREE/PD

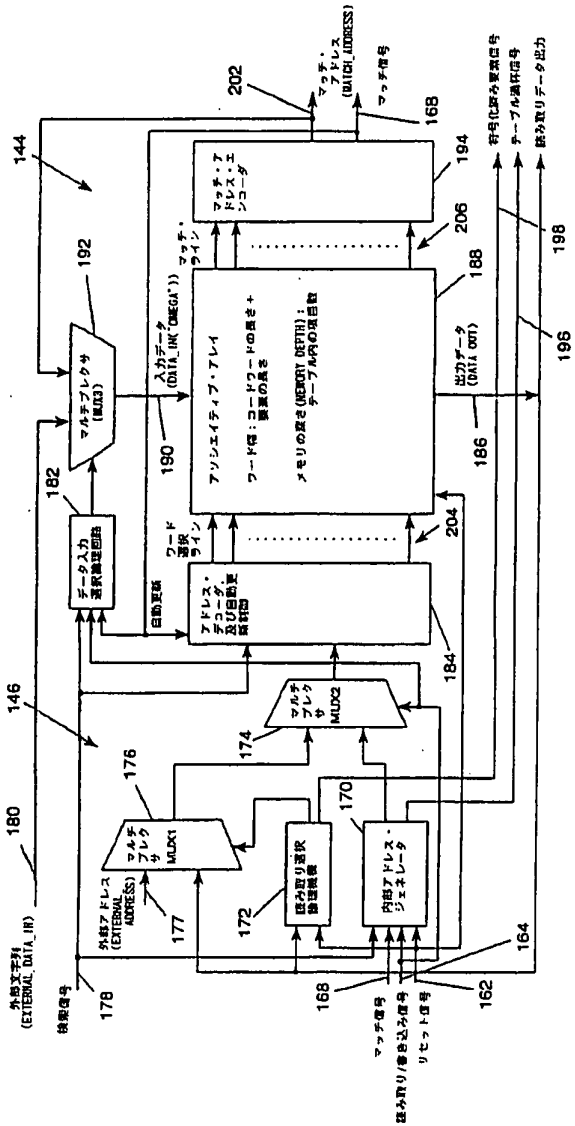
【図１９】



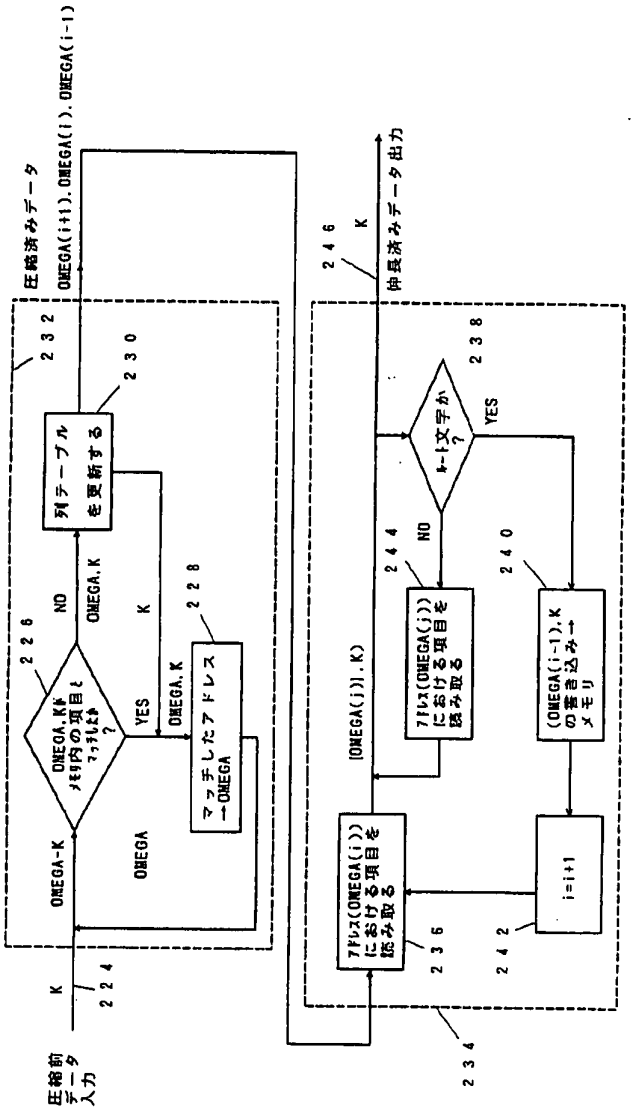
【図２０】



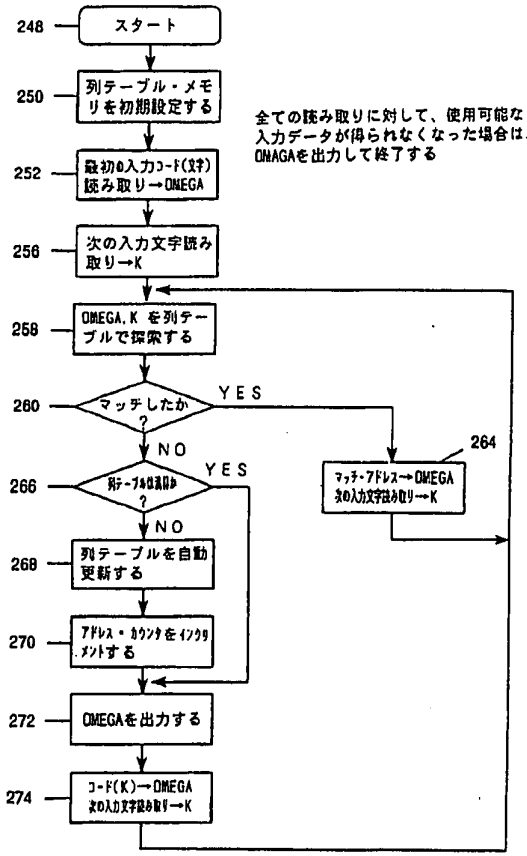
【図５】



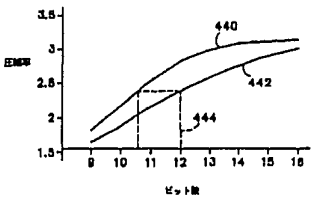
【図７】


圧縮済みデータ
OMEGA(i+1), OMEGA(i), OMEGA(i-1)
232
230
列テーブルを更新する
226
OMEGA, Kがメモリ内の項目とマッチしたか？
NO
OMEGA, K
YES
K
228
OMEGA, K
マッチしたアドレス→OMEGA
OMEGA
OMEGA-K
K
224
圧縮前データ入力
246
K
伸長済みデータ出力
238
ルート文字か？
YES
NO
244
アドレス(OMEGA(j))における項目を読み取る
240
(OMEGA(i-1)), Kの書き込みメモリ
[OMEGA(j)], K
アドレス(OMEGA(j))における項目を読み取る
236
i=i+1
242
234

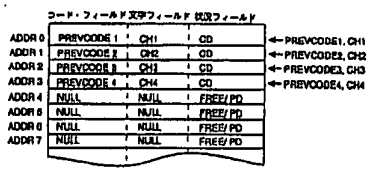
【図８】


248
スタート
250
列テーブル・メモリを初期設定する
252
最初の入力コード(文字)読み取り→OMEGA
256
次の入力文字読み取り→K
258
OMEGA, K を列テーブルで探索する
260
マッチしたか？
YES
NO
264
マッチ・アドレス→OMEGA次の入力文字読み取り→K
266
列テーブルは満杯か？
YES
NO
268
列テーブルを自動更新する
270
アドレス・カウンタをインクリメントする
272
OMEGAを出力する
274
コード(K)→OMEGA次の入力文字読み取り→K
全ての読み取りに対して、使用可能な入力データが得られなくなった場合は、ONAGAを出力して終了する

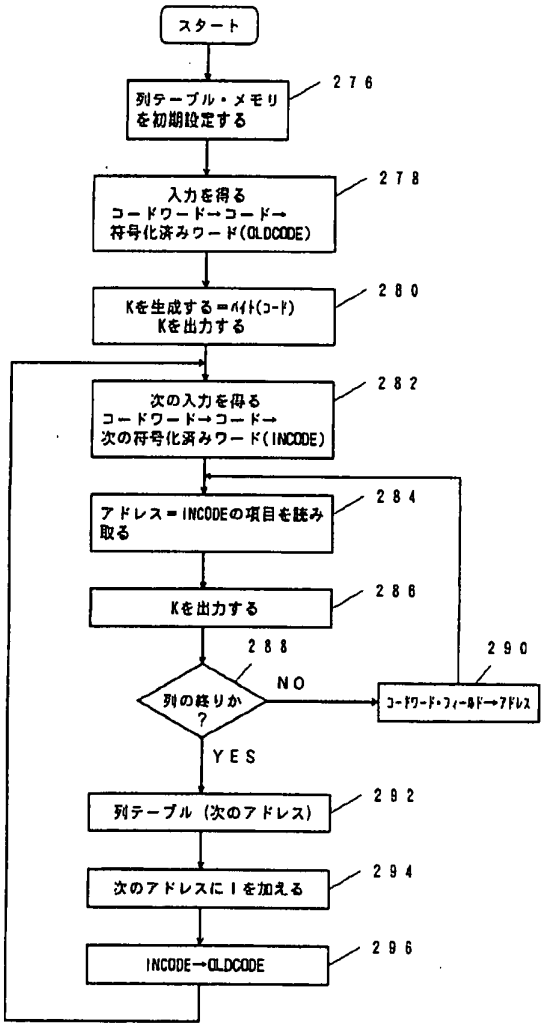
【図２１】


3.5
3
2.5
2
1.5
9
10
11
12
13
14
15
16
440
442
444

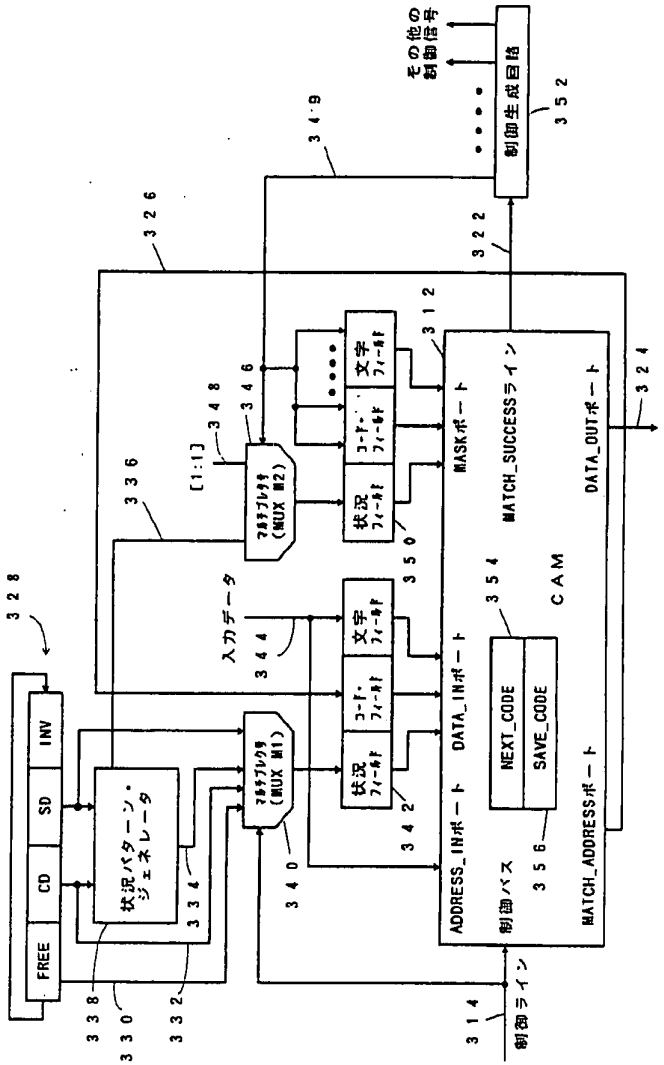
【図２２】


コード・フィールド
文字フィールド
状況フィールド
ADDR 0
ADDR 1
ADDR 2
ADDR 3
ADDR 4
ADDR 5
ADDR 6
ADDR 7
PREVCODE 1
PREVCODE 2
PREVCODE 3
PREVCODE 4
NULL
CH1
CH2
CH3
CH4
CD
FREE/PD
PREVCODE1, CH1
PREVCODE2, CH2
PREVCODE3, CH3
PREVCODE4, CH4

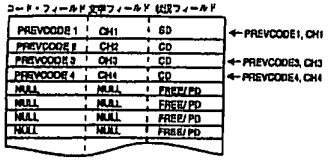
【図９】


スタート
列テーブル・メモリを初期設定する
276
入力を得るコードワード→コード→符号化済みワード(OLDCODE)
278
Kを生成する＝バイト(コード)Kを出力する
280
次の入力を得るコードワード→コード→次の符号化済みワード(INCODE)
282
アドレス＝INCODEの項目を読み取る
284
Kを出力する
286
列の終りか？
288
NO
コードワード・フィールド→アドレス
290
YES
列テーブル（次のアドレス）
292
次のアドレスに１を加える
294
INCODE→OLDCODE
296

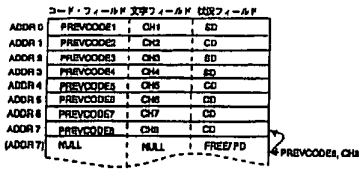
【図１６】


その他の制御信号
制御生成回路
352
349
326
322
312
MASKポート
MATCH_SUCCESSライン
DATA_OUTポート
324
336
[1:1]
348
346
マルチプレクサ (MUX M2)
文字フィールド
コードフィールド
状況フィールド
350
328
入力データ
344
INV
SD
CD
FREE
状況パターン・ジェネレータ
334
マルチプレクサ (MUX M1)
340
342
ADDRESS_INポート
DATA_INポート
354
NEXT_CODE
SAVE_CODE
CAM
制御バス
356
MATCH_ADDRESSポート
338
330
332
314
制御ライン

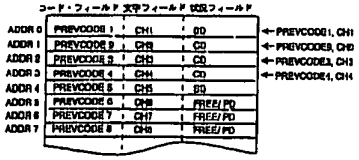
【図２３】



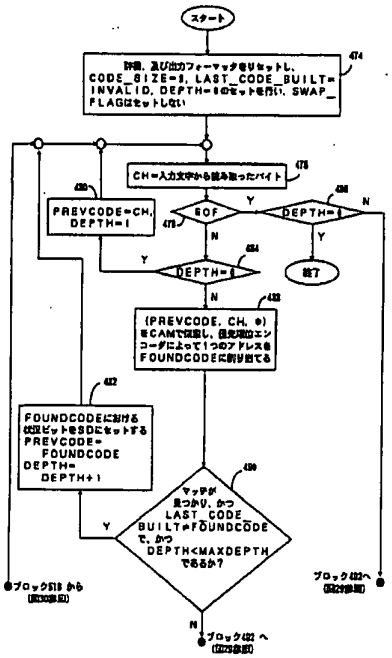
【図２４】



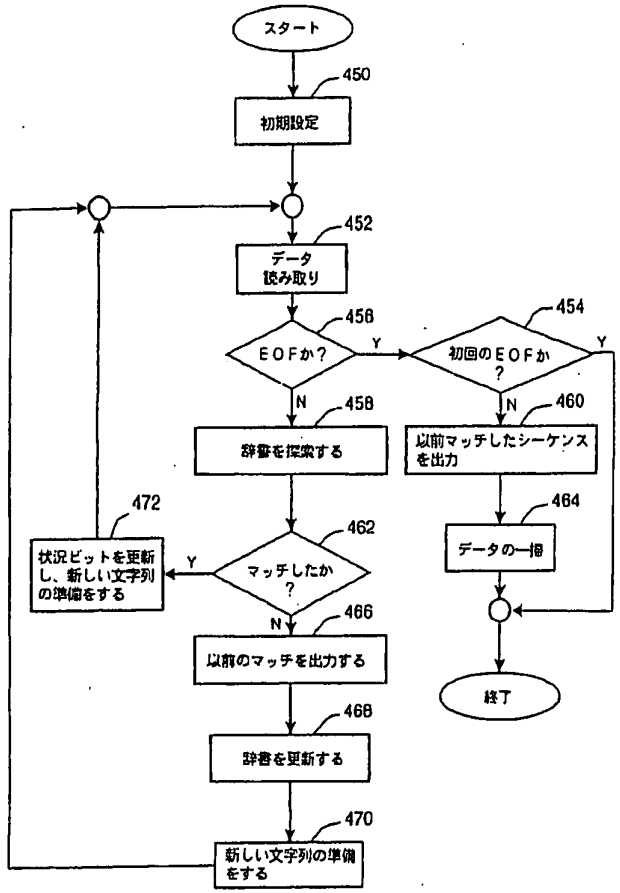
【図２６】



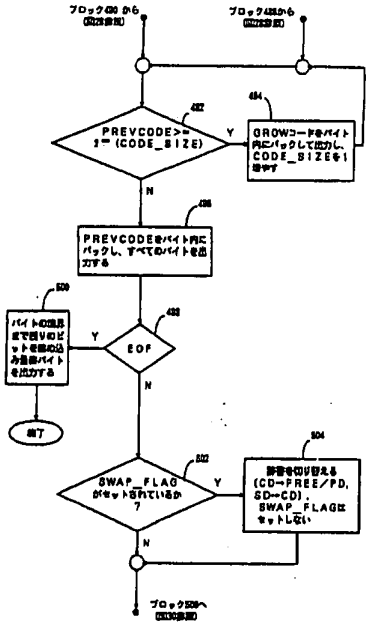
【図２８】



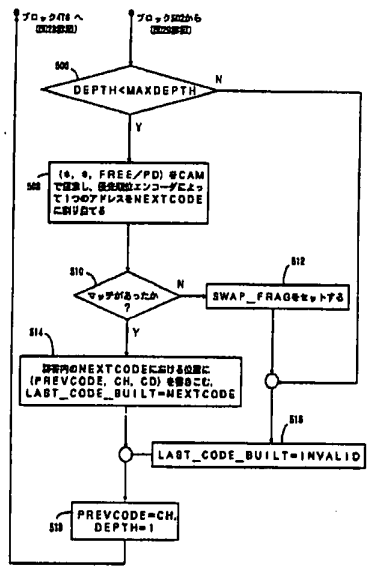
【図２７】



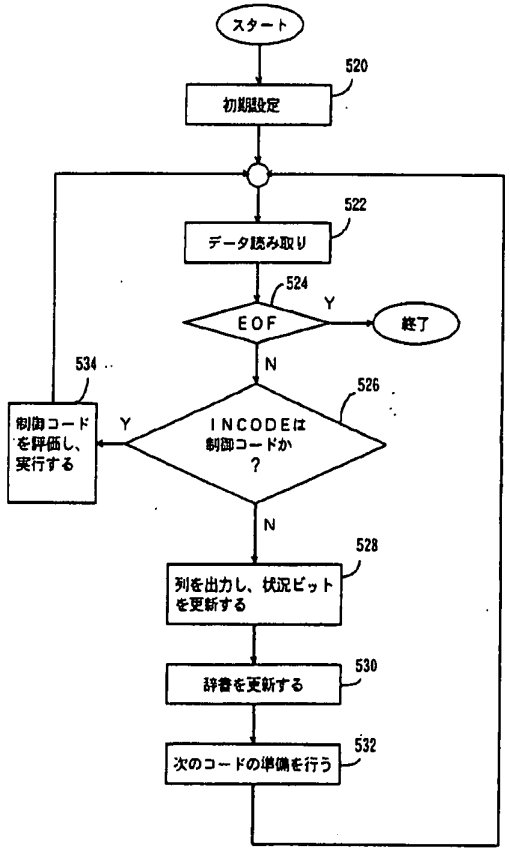
【図２９】


ブロック430 から
ブロック434から
PREVCODE>=
1<< (CODE_SIZE)
Y
GROWコードをバイト
内にパックして出力し、
CODE_SIZEを1
増やす
N
PREVCODEをバイト内に
パックし、すべてのバイトを出
力する
バイト内の情報
までを1つのビ
ットを除いた
お最後のバイト
を出力する
Y
EOF
N
終了
SWAP_FLAG
がセットされているか
?
Y
辞書を切り替える
(CD→FREE/PD,
SD→CD)、
SWAP_FLAGは
セットしない
N
ブロック520へ

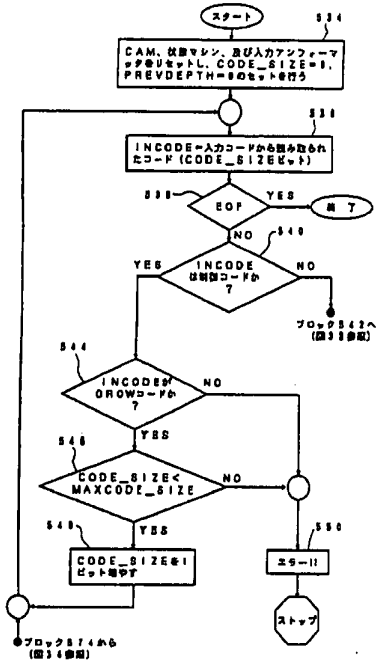
【図３０】


ブロック478 へ
ブロック502から
DEPTH<MAXDEPTH
N
Y
(0, 0, FREE/PD) をCAM
で探索し、優先順位エンコーダによっ
て1つのアドレスをNEXTCODE
に割り当てる
マッチがあったか
?
N
SWAP_FRAGをセットする
Y
辞書内のNEXTCODEに当たる位置に
(PREVCODE, CH, CD) を書きこむ、
LAST_CODE_BUILT=NEXTCODE
LAST_CODE_BUILT=INVALID
PREVCODE=CH,
DEPTH=1

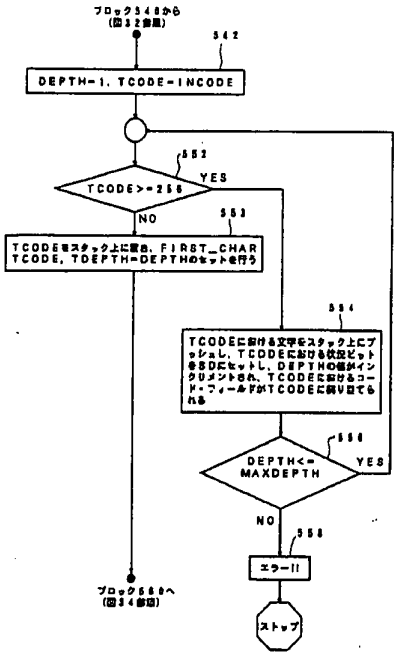
【図３１】


スタート
初期設定
データ読み取り
EOF
Y
終了
N
INCODEは
制御コードか
?
Y
制御コード
を評価し、
実行する
N
列を出力し、状況ビット
を更新する
辞書を更新する
次のコードの準備を行う

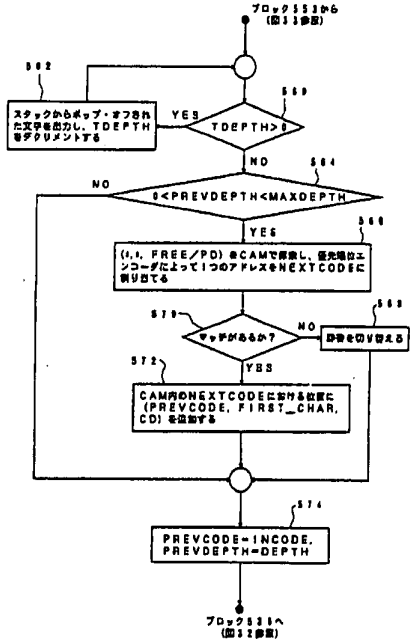
【図３２】



【図３３】



【図３４】



フロントページの続き

(72)発明者　ガディエル・セローシ
アメリカ合衆国カリフォルニア州95014ク
パーティノ，ミルキー・ウエイ・1123